THESIS

EXTENSIONS TO THE ICALENDAR DATA MODEL: HIERARCHICAL SCHEDULES AND TEMPORALLY RELATED EVENTS

by

Lesley W. Chiu

September 2012

Thesis Co-Advisors:	Thomas Otani
	Mantak Shing

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2012	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Extensions to the iCalendar Data Model: Hierarchical Schedules and Temporally Related Events			**5. FUNDING NUMBERS**
6. AUTHOR(S) Lesley W. Chiu			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ______N/A______.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Computer calendaring has matured to the point where people's calendars are easily synchronized across devices, shared between other people and organizations, and used to coordinate and schedule mutually agreeable meeting times. These and other advances have made computer calendaring indispensable in people's personal and professional lives.

Despite the recent advances in the scheduling capabilities of computer calendaring software, there are still areas of scheduling that are not so well accomplished—or even possible—in today's computer calendaring state of the art. Non-fixed date template schedules, used in military exercise and contingency plans (e.g., POA&M), are examples of schedules that are highly flexible and reusable. Because of their unique scheduling properties and relationships, however, they cannot be modeled in mainstream computer calendaring tools. As a result, when applying these types of schedules to a computer calendar, a great deal of human calculation, manual manipulation, and data entry are required to get the schedule information into the calendar.

This thesis explores extensions to the iCalendar data model. These extensions will enable schedule hierarchies and temporally related events to be modeled in mainstream calendaring applications. The results will show considerable savings in human processing effort and that reduction in data-entry errors is possible.

14. SUBJECT TERMS Calendaring, Calendaring Extensions, Temporally Related Events, Hierarchical Calendar Structure			**15. NUMBER OF PAGES** 103
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

EXTENSIONS TO THE ICALENDAR DATA MODEL: HIERARCHICAL SCHEDULES AND TEMPORALLY RELATED EVENTS

Lesley W. Chiu
Lieutenant Colonel, United States Marine Corps
B.S., University of California at San Diego, 1991

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2012

Author:	Lesley W. Chiu
Approved by:	Thomas Otani Thesis Co-Advisor
	Mantak Shing Thesis Co-Advisor
	Peter J. Denning Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Computer calendaring has matured to the point where people's calendars are easily synchronized across devices, shared between other people and organizations, and used to coordinate and schedule mutually agreeable meeting times. These and other advances have made computer calendaring indispensable in people's personal and professional lives.

Despite the recent advances in the scheduling capabilities of computer calendaring software, there are still areas of scheduling that are not so well accomplished—or even possible—in today's computer calendaring state of the art. Non-fixed date template schedules, used in military exercise and contingency plans (e.g., POA&M), are examples of schedules that are highly flexible and reusable. Because of their unique scheduling properties and relationships, however, they cannot be modeled in mainstream computer calendaring tools. As a result, when applying these types of schedules to a computer calendar, a great deal of human calculation, manual manipulation, and data entry are required to get the schedule information into the calendar.

This thesis explores extensions to the iCalendar data model. These extensions will enable schedule hierarchies and temporally related events to be modeled in mainstream calendaring applications. The results will show considerable savings in human processing effort and that reduction in data-entry errors is possible.

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
CS	Computer Science
DE	Dependent Event
EE	End-to-End Relationship
ES	End-to-Start Relationship
HTTP	Hyper Text Transfer Protocol
IA	Information Assurance
IE	Independent Event
IETF	Internet Engineering Task Force
IMC	Internet Mail Consortium
MIME	Multipurpose Internet Mail Extensions
NE	Network Engineering
PC	Planning Committee
PDA	Personal Digital Assistant
PDI	Personal Data Interchange
POA&M	Plan of Action and Milestones
PM	Project Management
RFC	Request For Comment
SE	Start-to-End Relationship
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SS	Start-to-Start Relationship
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I started this program not knowing much about computer science. I know a bit more now, but most of all I know I am humbled by all the great minds who have pioneered this field, and those who continue to advance it.

I would like to thank the NPS Computer Science faculty for expanding the dimensions of my brain by at least a base-2 order of magnitude. Also, thanks to my fellow computer science peers, especially to LT Joey Carter (USN), Major Jon Petersen (USMC), and Major Tarrell Giersch (USMC). Many team projects and long hours behind the keyboard or white-board were made easier with your insight and humor. A special thanks goes to my advisors for their patience and guidance throughout this thesis work.

Most of all, I would like to thank my family for all their support and sacrifice. I hope I can return the favor in some fashion later in life.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Computer-based calendaring and scheduling is one of the most widely used and mature examples of cloud-based collaboration in which users want to share a common picture or perspective across multiple devices and with other users. With the development and mainstream adoption of networked handheld computing devices (personal digital assistants (PDA) and smartphones) users have embraced the ability to easily share and synchronize their schedules with other users and devices. Because of this, the reliance and dependency of these electronic calendars are increasing for people both in their personal and professional lives.

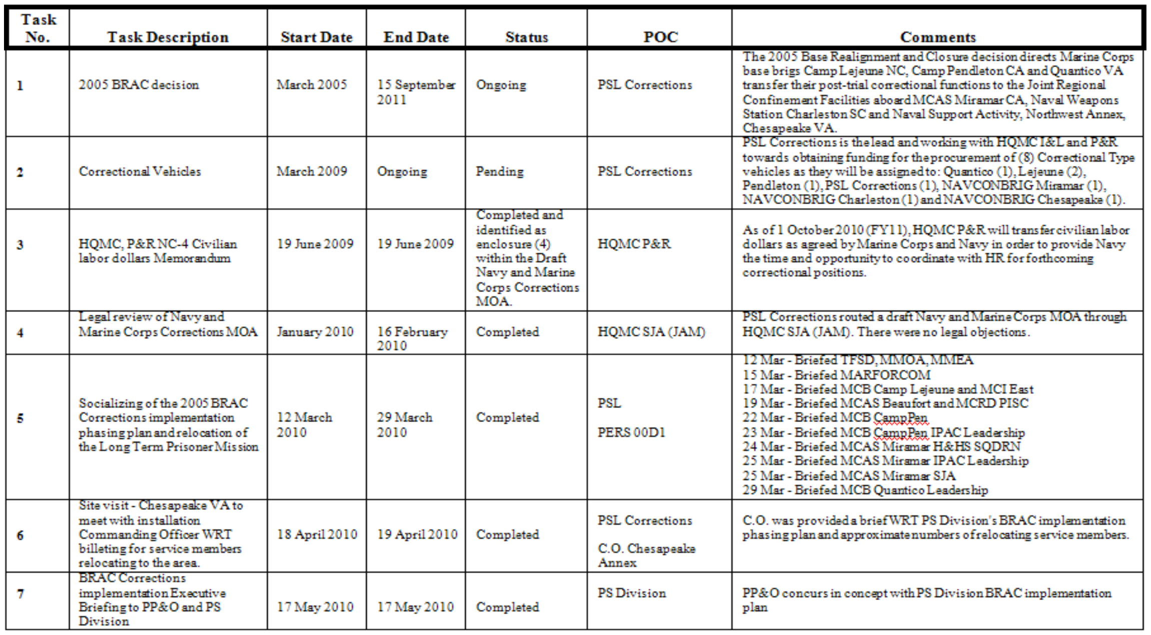
Although electronic calendars have evolved considerably, they are still limited to the basic capability of scheduling singular events. While this may be suitable for individuals and their basic scheduling needs, it is inadequate for organizations and their need for more robust planning and scheduling tools. Since organizations are hierarchical, so are their schedules. Organizational planning requires the ability to assign tasks to individuals and groups. These identified and assigned tasks are not independent from one another. Likely, one identified task belonging to one group must precede or follow the identified task of another. Thus, in project planning, one group’s schedule is not entirely independent from the others in the organization and indeed many scheduled events are essentially a series of relatable events. It is these temporal relationships in such series that cannot be captured in today’s mainstream calendaring systems.

Mainstream calendaring systems are limited in two fundamental ways to better support organizational planning. First, calendaring systems do not allow for hierarchical relationships between schedules. Currently, calendaring systems allow multiple schedules (calendars) to be created but only in a flat, unstructured way. This current state is adequate for an individual or a small organization’s functional schedule breakdown (e.g., an individual may have schedules for home, work, school), but is inadequate to model the related schedules of a large-hierarchical organization.

Secondly, calendaring systems today do not allow for temporal dependencies between events. In planned projects, events are often related with other events temporally (i.e., the start of one event occurs some time after the completion of another). Although these individual events can be specified in today's calendaring systems, their sequencing relationships cannot. This lack of sequencing support is a big limitation to using mainstream calendaring tools for project planning.

B. ILLUSTRATION OF COMPLEX ORGANIZATIONAL PLANNING

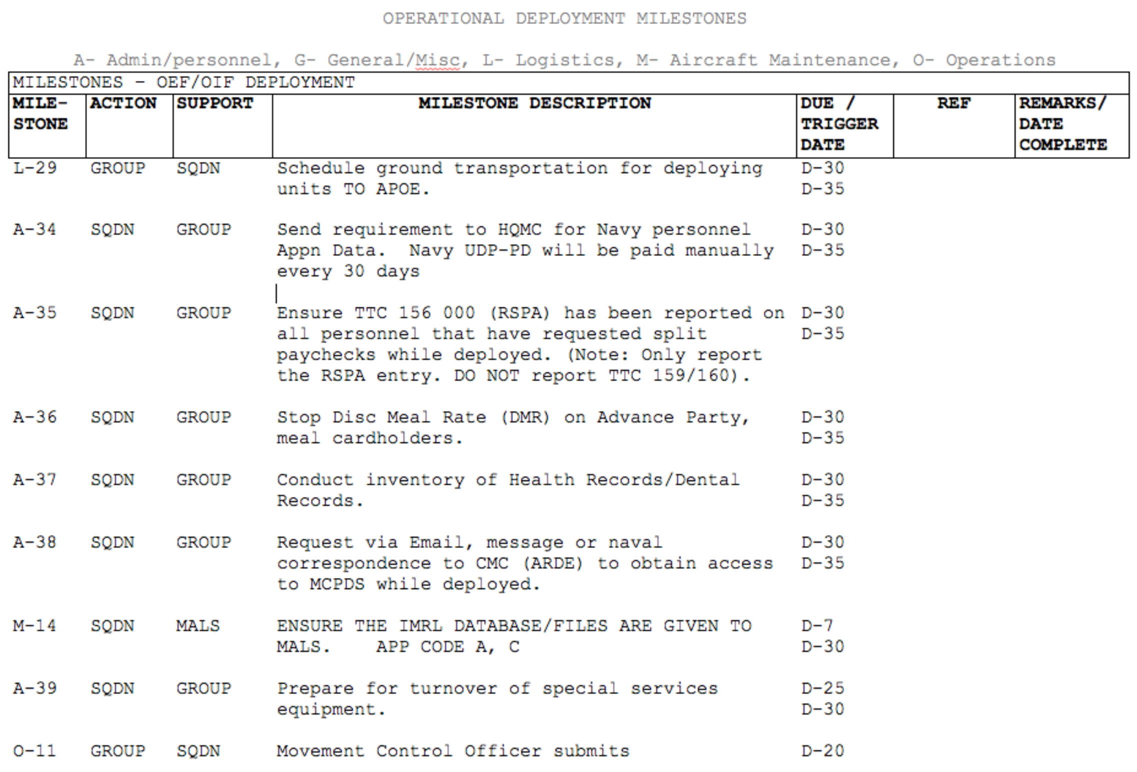
A Plan of Action and Milestones (POA&M) is a planning tool widely used throughout the Department of Defense and industry to outline groups of interdependent tasks to accomplish a larger event or goal. The POA&M template is used for acquisition programs, military exercise planning, unit deployment procedures, and a host of other applications. It is a planning tool to identify action-items necessary to complete a project and track major milestones for project completion. They list the tasks, task owners, relative time frame for start and/or completion, and identify any dependencies from one task to another. They can include a hierarchy of tasks either grouped by functional area or by functional task. Figure 1 shows an example of a POA&M with specified dates for start and completion of each task.



Task No.	Task Description	Start Date	End Date	Status	POC	Comments
1	2005 BRAC decision	March 2005	15 September 2011	Ongoing	PSL Corrections	The 2005 Base Realignment and Closure decision directs Marine Corps base brigs Camp Lejeune NC, Camp Pendleton CA and Quantico VA transfer their post-trial correctional functions to the Joint Regional Confinement Facilities aboard MCAS Miramar CA, Naval Weapons Station Charleston SC and Naval Support Activity, Northwest Annex, Chesapeake VA.
2	Correctional Vehicles	March 2009	Ongoing	Pending	PSL Corrections	PSL Corrections is the lead and working with HQMC I&L and P&R towards obtaining funding for the procurement of (8) Correctional Type vehicles as they will be assigned to: Quantico (1), Lejeune (2), Pendleton (1), PSL Corrections (1), NAVCONBRIG Miramar (1), NAVCONBRIG Charleston (1) and NAVCONBRIG Chesapeake (1).
3	HQMC, P&R NC-4 Civilian labor dollars Memorandum	19 June 2009	19 June 2009	Completed and identified as enclosure (4) within the Draft Navy and Marine Corps Corrections MOA.	HQMC P&R	As of 1 October 2010 (FY11), HQMC P&R will transfer civilian labor dollars as agreed by Marine Corps and Navy in order to provide Navy the time and opportunity to coordinate with HR for forthcoming correctional positions.
4	Legal review of Navy and Marine Corps Corrections MOA	January 2010	16 February 2010	Completed	HQMC SJA (JAM)	PSL Corrections routed a draft Navy and Marine Corps MOA through HQMC SJA (JAM). There were no legal objections.
5	Socializing of the 2005 BRAC Corrections implementation phasing plan and relocation of the Long Term Prisoner Mission	12 March 2010	29 March 2010	Completed	PSL PERS 00D1	12 Mar - Briefed TFSD, MMOA, MMEA 15 Mar - Briefed MARFORCOM 17 Mar - Briefed MCB Camp Lejeune and MCI East 19 Mar - Briefed MCAS Beaufort and MCRD PISC 22 Mar - Briefed MCB CampPen 23 Mar - Briefed MCB CampPen IPAC Leadership 24 Mar - Briefed MCAS Miramar H&HS SQDRN 25 Mar - Briefed MCAS Miramar IPAC Leadership 25 Mar - Briefed MCAS Miramar SJA 29 Mar - Briefed MCB Quantico Leadership
6	Site visit - Chesapeake VA to meet with installation Commanding Officer WRT billeting for service members relocating to the area.	18 April 2010	19 April 2010	Completed	PSL Corrections C.O. Chesapeake Annex	C.O. was provided a brief WRT PS Division's BRAC implementation phasing plan and approximate numbers of relocating service members.
7	BRAC Corrections implementation Executive Briefing to PP&O and PS Division	17 May 2010	17 May 2010	Completed	PS Division	PP&O concurs in concept with PS Division BRAC implementation plan

Figure 1. POA&M Example with Fixed Dates. From [1].

Many POA&Ms are pre-prepared for a specific purpose. For contingency operations, POA&Ms serve as a pre-planned template of action. These template plans are no different than a specific POA&M, such as that in Figure 1, except that the dates for start and completion are relative to an overall trigger event. Figure 2 demonstrates this idea. This POA&M template is for the operational deployment of a Marine Aircraft Wing. The variable *D* in the *Due/Trigger Date* column stands for the deployment date. Once the order for deployment has been given, and the deployment date is known, the individual milestones will have specified dates for completion. These types of templates are flexible and easily reusable. The temporal nature between events is specified and the overall schedule can be easily applied to an actual deployment by specifying the trigger event.



OPERATIONAL DEPLOYMENT MILESTONES

A- Admin/personnel, G- General/Misc, L- Logistics, M- Aircraft Maintenance, O- Operations

MILESTONES – OEF/OIF DEPLOYMENT

MILE-STONE	ACTION	SUPPORT	MILESTONE DESCRIPTION	DUE / TRIGGER DATE	REF	REMARKS/ DATE COMPLETE
L-29	GROUP	SQDN	Schedule ground transportation for deploying units TO APOE.	D-30 D-35		
A-34	SQDN	GROUP	Send requirement to HQMC for Navy personnel Appn Data. Navy UDP-PD will be paid manually every 30 days	D-30 D-35		
A-35	SQDN	GROUP	Ensure TTC 156 000 (RSPA) has been reported on all personnel that have requested split paychecks while deployed. (Note: Only report the RSPA entry. DO NOT report TTC 159/160).	D-30 D-35		
A-36	SQDN	GROUP	Stop Disc Meal Rate (DMR) on Advance Party, meal cardholders.	D-30 D-35		
A-37	SQDN	GROUP	Conduct inventory of Health Records/Dental Records.	D-30 D-35		
A-38	SQDN	GROUP	Request via Email, message or naval correspondence to CMC (ARDE) to obtain access to MCPDS while deployed.	D-30 D-35		
M-14	SQDN	MALS	ENSURE THE IMRL DATABASE/FILES ARE GIVEN TO MALS. APP CODE A, C	D-7 D-30		
A-39	SQDN	GROUP	Prepare for turnover of special services equipment.	D-25 D-30		
O-11	GROUP	SQDN	Movement Control Officer submits	D-20		

Figure 2. POA&M Template Example with Non-Specified Dates. From [2].

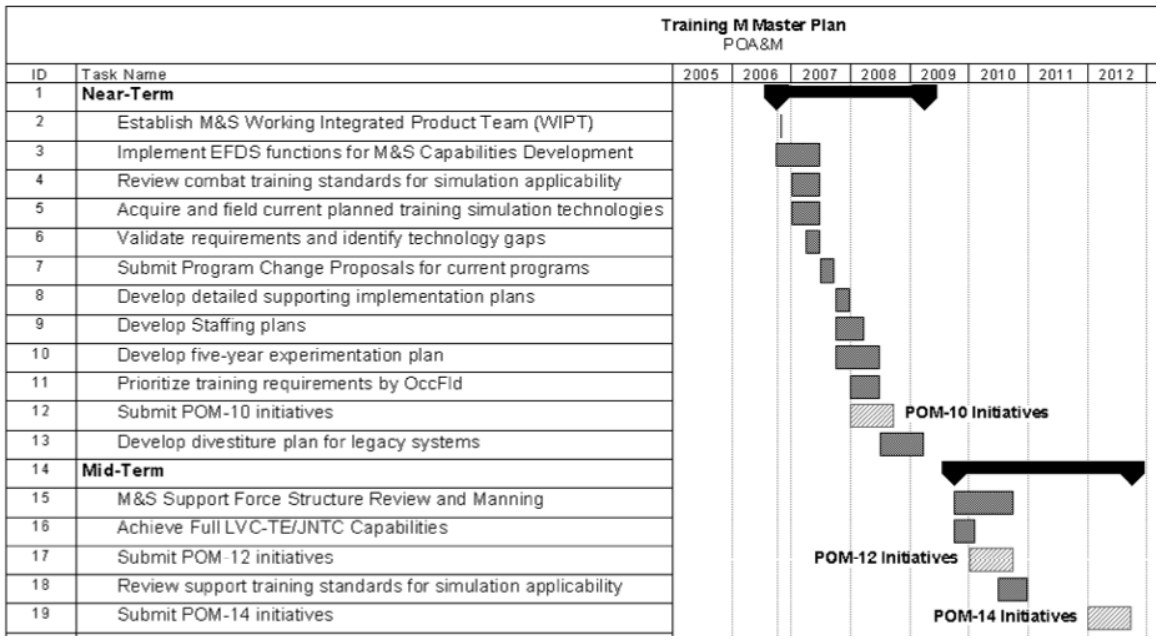
C. POA&M IMPLEMENATION IN ELECTRONIC CALENDARS

POA&M's can be created and managed by sophisticated software such as MS Project™, but usually are constructed in table format in software such as MS Word™ or Excel™. Only in large-scale acquisition, design, or construction projects is something as powerful as MS Project used. Such project planning software is necessary when detailed planning and cost estimation are a necessity. Due to their added expense and complexity though, the use of such project planning software is often too much for organizations and individuals not involved in formal project management disciplines.

For the average user, capturing the essential tasks, subdivision of work, critical dates, and milestones is easily and naturally accomplished in a table format. Mainstream word document and spreadsheet applications are more than adequate to do this. Once created using these tools, POA&Ms are easily shared and communicated. Unfortunately, however, using word processors and spreadsheets means that the POA&M is more of a passive guide or outline, not an active one.

Because the POA&M is a planning, scheduling, and tracking tool, it is desirable to have this schedule information in a calendar. Calendars are an active planning tool. People access them often, sometimes several times a day. Time-space relationships are more easily seen in a calendar. Scheduling conflicts can be more easily identified when separate schedules are applied to a single calendar.

A POA&M with specified dates such as in Figure 1 is easily put in a calendar. Each task is entered as an event with specified start and end times. Depending on the complexity of the POA&M and the number of assignable actors involved, different calendars can be created for each actor, each visually identifiable from one another. Once input in the calendar, various views of the POA&M could be generated (e.g., monthly, weekly, daily, and agenda). One considerably useful view found in project management software (but not in calendaring software, even though it can be realized with the information entered in the calendar) is a Gantt chart depicted in Figure 3.



ID	Task Name
1	**Near-Term**
2	Establish M&S Working Integrated Product Team (WIPT)
3	Implement EFDS functions for M&S Capabilities Development
4	Review combat training standards for simulation applicability
5	Acquire and field current planned training simulation technologies
6	Validate requirements and identify technology gaps
7	Submit Program Change Proposals for current programs
8	Develop detailed supporting implementation plans
9	Develop Staffing plans
10	Develop five-year experimentation plan
11	Prioritize training requirements by OccFld
12	Submit POM-10 initiatives
13	Develop divestiture plan for legacy systems
14	**Mid-Term**
15	M&S Support Force Structure Review and Manning
16	Achieve Full LVC-TE/JNTC Capabilities
17	Submit POM-12 initiatives
18	Review support training standards for simulation applicability
19	Submit POM-14 initiatives

Figure 3. POA&M in Differing Calendar Perspectives. From [3].

However, representing a POA&M template in a calendar, such as the one described in Figure 2, is not possible without specifying fixed dates. In this example, the variable *D* (deployment date) needs to be specified and from that the remaining dates can be determined. Once the deployment date is known, the calendar can be populated with the tasks and milestones laid out in the POA&M. What if the deployment date shifts? Or what if only a tentative date is given but the actual date is still to be determined? In these cases, the calendar owner must painstakingly adjust each task and milestone relative to the new deployment date. This may be no trivial task, depending on the number of tasks/milestones involved.

If the offset relationships could be captured in the calendar (i.e., the task "schedule ground transportation for deploying units to APOE"'—which is to occur 30 days prior to the deployment date), then the corrected dates for all of the temporally related calendar events could be automatically adjusted when changes to the deployment date are made. Calendaring systems today are not capable of this type of event modeling, and therefore do not ideally support representation of a dynamic or reusable POA&M.

Simple extensions to the calendaring (open-source) data and conceptual model would allow developers to include the functionality to specify these temporal relationships. If this were to occur, the electronic calendar would easily augment and enhance the POA&M process.

D. ORGANIZATION OF THESIS

Chapter II provides background information for project management and scheduling. It also discusses the current state of the art of mainstream calendaring systems by outlining common features and capabilities. Also the iCalendar data format, a formal grammar used for calendar storage and data exchange, is presented. This is the basis for which the set of proposed extensions would be built from.

Chapter III describes the requirements for the proposed set of extensions by discussing how such a system can be used to better model these scheduling relationships and hierarchies.   Stakeholders and Actors are identified and defined. System features and functionality, and essential use cases are presented as formal requirements.

Chapter IV proposes a conceptual data and application model for the proposed calendaring system.

Chapter V shows the benefits of these calendaring extensions by providing a case study and applying selected use cases described in Chapter III. In this context, a comparison between the two methodologies will reveal the benefit and utility these extensions will provide.

Chapter VI concludes the thesis and provides recommendations for future research and development.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. PLANNING, SCHEDULING, AND PROJECT MANAGEMENT

1. Planning and Scheduling

Planning is the process of satisfying requirements and future objectives through the assignment of resources for future action. According to [4], planning seeks to answer the following questions:

1) What product or service is to be provided?

2) On what scale will it be provided?

3) What resources will be made available?

Scheduling on the other hand, is the process of defining and modifying events often for the purpose of objective realization, resource allocation, and constraint management. Scheduling involves applying people, units, or resources to events and/or locations and seeks to answer the following questions:

1) Which resources will be allocated to perform each task?

2) When will each task be performed?

Planning and scheduling permeate human behavior. We may do it informally (i.e., without formal representation or tools)—e.g., making a mental note to mow the lawn Saturday morning. More often, though, we associate planning and scheduling as a more deliberate process of collaboration through the use of planning techniques and scheduling tools (e.g., a major military spring offensive).

The primary artifact resulting from the planning and scheduling process is the *schedule*—a document of planned order and timed events created through the process of planning and scheduling. Bus schedule, conference, and television schedules are examples of documents containing planned, timed events.

2. Sequencing

When constraints are placed on the order of execution of events (i.e., Task A must be completed before Task B can begin) the notion of the order of events is referred to as sequencing. Sequencing can be explicitly outlined in a schedule. For example, the outline of a bus schedule shows the time and location (bus stop) that the bus will make throughout the day. Other schedules may not contain such specific ordering and are not clear in establishing whether a sequential relationship exists between events. For example, examination of someone's personal schedule might indicate they will make a bank stop at 3:30–3:45 p.m. and will be at the post office from 4:00–4:15 p.m. From the information as presented, there is no way to tell if the bank stop must precede the postal stop or if the order of the two events can be interchanged. It may turn out in fact that the trip to the bank must be made first because the person's intent is to obtain a cashier's check and mail it.

This type of sequencing or constraint modeling cannot be achieved in a mainstream calendaring application. No enforcement mechanism or indication that the bank visit must precede the post office visit (with the exception of specifying the need to visit the bank in the notes or remarks section of the post office event) can be imposed. Large-scale projects have many such sequencing relationships. Without a way to model sequencing relationships in a scheduling tool, such as an electronic calendar, the potential for unintended consequences increases as scheduling changes occur. In the previous example, the onus remains on the calendar user to make sure not to go to the post office first. This may seem obvious and intuitive, but when the people who execute the schedule differ from the ones who originally wrote it, the sequencing relationships and constraints may not be apparent and unintended consequences can develop as a result.

3. Project Management

Project Management (PM) is defined as a temporary endeavor undertaken to produce a unique product, service, or result in a defined timeframe and cost. A problem scheduled for solution [5]. PM is considered by many to be its own class of management due to the uniqueness and specificity of what is being accomplished. Planning and scheduling are major facets of PM as is budgeting and cost estimation. For planning and scheduling, the calendar is the primary tool.

PM mandates effective: communications, cooperation, teamwork, and trust [6]. The calendar is one of the main communication tools for PM. It is central to one of the main constraints for a project—time, and indirectly linked to the financial constraint of the project—cost.

PM software tools are very capable and offer features such as: Planning, Tracking, and Monitoring; Reports (budget, cost, earned value analysis, critical path analysis), Project calendar, What-if analysis, Multi-project analysis. These capabilities are essential to a large and costly project. They offer the capability to plan and measure the progress of the project to ensure the project comes in on time and on budget. For those large-scale projects where tight control and detailed integration of activities, schedules, and cost are necessary, dedicated PM software becomes essential.

For many individuals and organizations though, PM software can be too complex or costly for use. An organization that is planning a weeklong conference, or a couple that are planning a wedding, or a military unit that is planning a deployment overseas, all need to approach their problem with a PM focus in mind. Planning and scheduling are vital aspects of these less than large-scale projects, but the integration and control that are inherent in PM software make them overkill for such use.

4. Representational Views of Schedules

Differing perspectives of a schedule are useful depending on the context in which it is viewed. A near-term perspective will require display of only those events that are immediately (i.e., minutes to hours) pending, whereas a long-term perspective will require display of those events covering a substantially greater period (i.e., weeks to months). Figures 4 through 6 represent these perspectives, or calendar views, which are implemented in most mainstream calendaring tools.

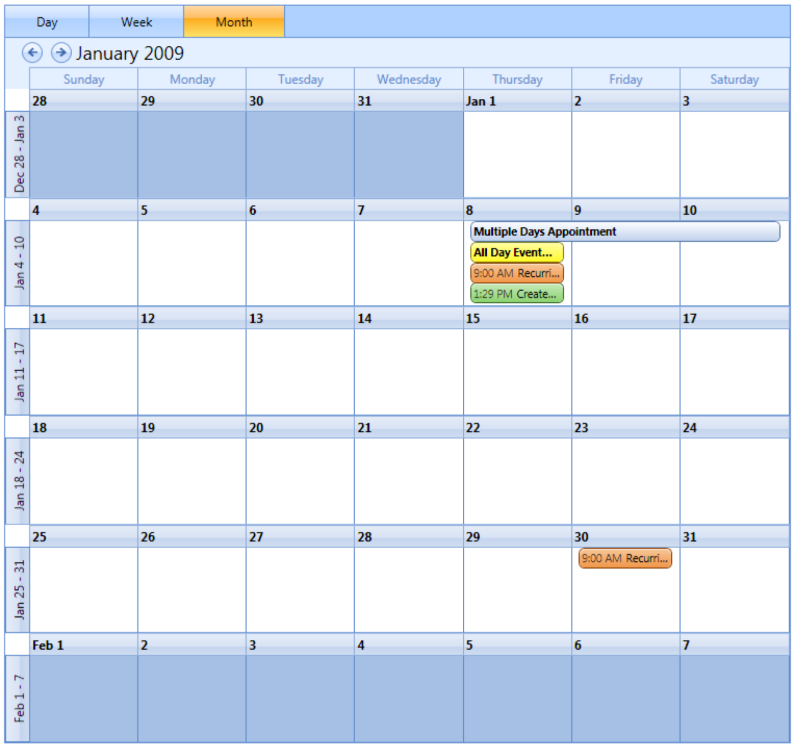


Figure 4. Monthly View. From [7].

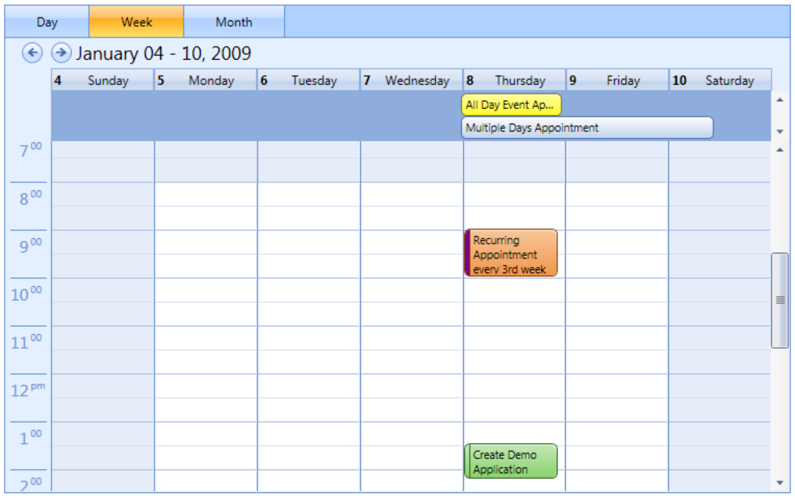


Figure 5. Weekly View. From [7].

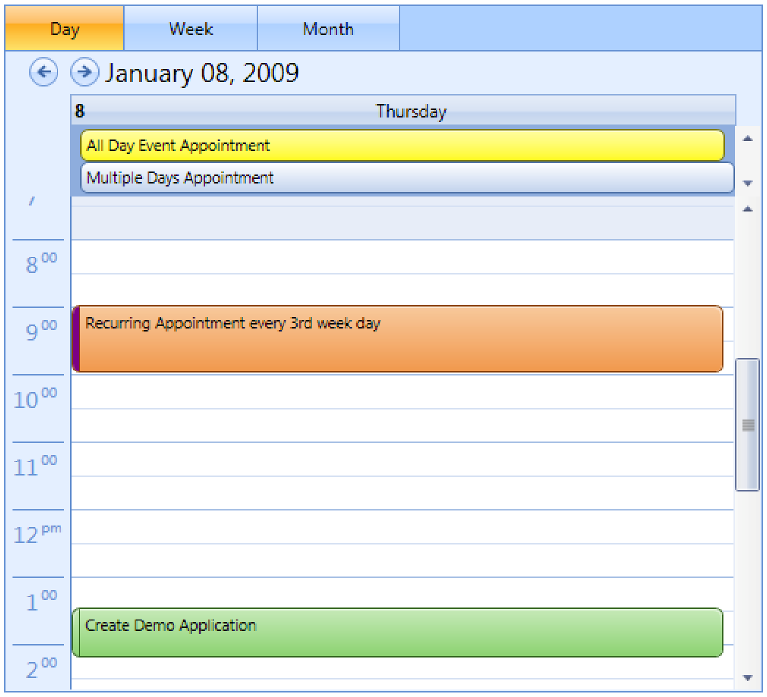
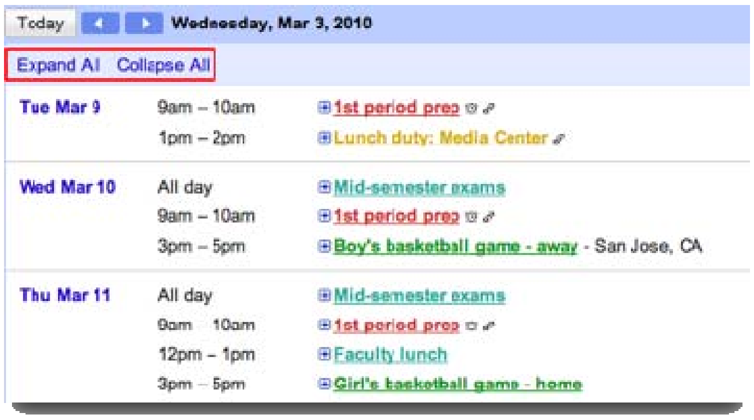


Figure 6. Daily View. From [7].

The list view (Figure 7) and timeline view (Figure 8), however, are mostly only found in project planning software.


Today Wednesday, Mar 3, 2010

Expand All Collapse All

Tue Mar 9	9am – 10am	⊞1st period prep
	1pm – 2pm	⊞Lunch duty: Media Center
Wed Mar 10	All day	⊞Mid-semester exams
	9am – 10am	⊞1st period prep
	3pm – 5pm	⊞Boy's basketball game - away - San Jose, CA
Thu Mar 11	All day	⊞Mid-semester exams
	9am – 10am	⊞1st period prep
	12pm – 1pm	⊞Faculty lunch
	3pm – 5pm	⊞Girl's basketball game - home

Figure 7. List (Agenda) View. From [8].

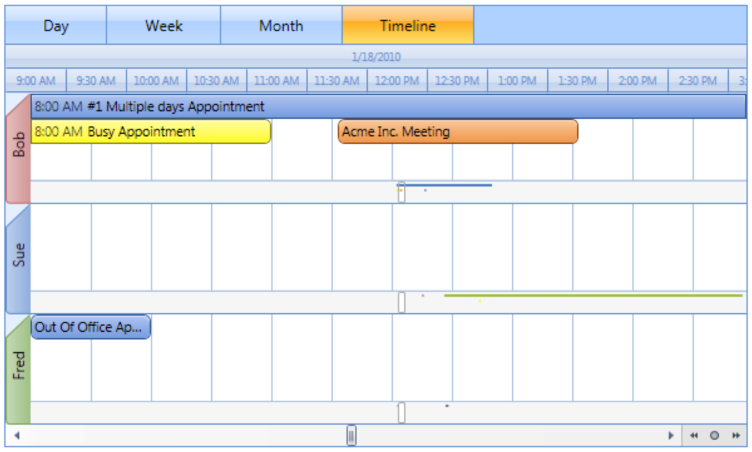


Figure 8. Timeline View (Gantt Chart). From [7].

Each of these calendar views can be applied to a schedule. Depending on the length of the schedule, some views may not be as useful as others. This is especially true for those views with fixed time scales (e.g., daily, weekly, and monthly). A monthly view for example, will not provide much information for a schedule that takes place in one 24-hour period. Other views, such as the timeline view, are scalable. The timeline can be "stretched" or "shrunk" as necessary to provide a timeframe interval. Still other views, such as the list view, have no scale and it is indeed difficult to realize the temporal relation between events without cognitively analyzing dates and times.

B. COMPUTER-BASED PLANNING AND SCHEDULING TOOLS

1. Project Management Software

When tight control and detail of schedule and cost are required, project management tools are necessary to plan, track, estimate, and budget the project. Project management software includes these capabilities and some include more such as resource identification and allocation, quality management, and communication and collaboration tools. The key benefit to using project management software is its tight integration primarily across resource scheduling and cost management.

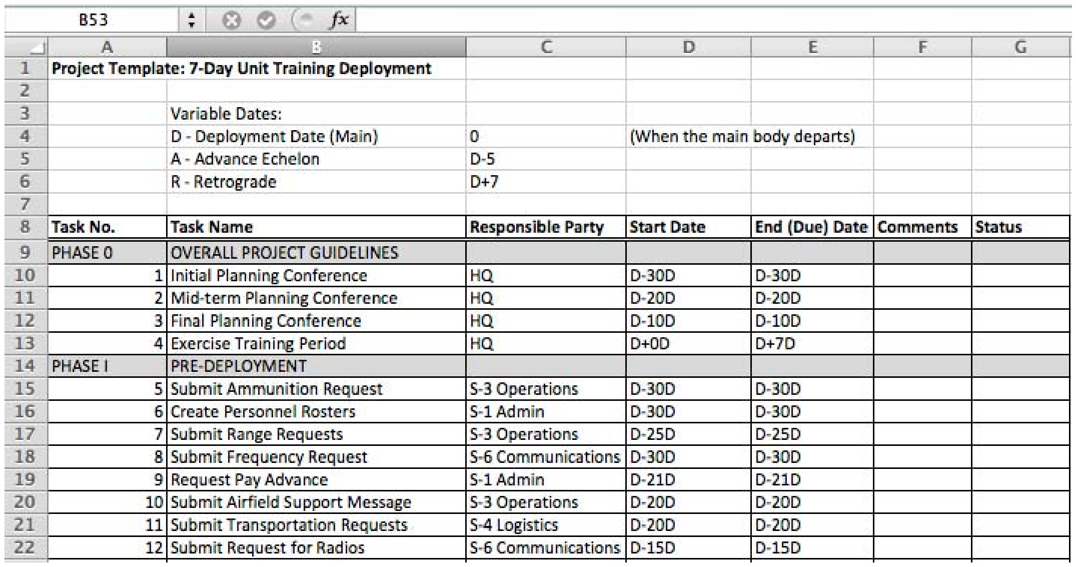
A number of private vendors offer project management software solutions, the most popular and well known is MS Project. There are also a number of open source software packages that are freely available. Some software solutions are meant to suit small-scale projects and might support only single user, single machine use. Others are more robust offering collaborative viewing and editing through web-based tools. Some even integrate with corporate e-mail and calendaring systems, yet others can integrate with billing and accounting systems.

Project management software generally offers robust scheduling tools and capabilities. Scheduling categories such as tasks, phases, and milestones can be delineated for particular events. Standard attributes such as start, end, and duration can be applied, and a variety of preformatted and custom attributes can be to events as well. Project management software also gives the ability to breakdown tasks into subtasks and produce hierarchical relationships necessary for a work breakdown structure.

2. Spreadsheets

Spreadsheet and other applications that allow tabular formatting (i.e., word processing programs) give users the ability to setup table layouts that support organization of information into rows and columns. Primarily developed to handle numeric computation, spreadsheet applications also allow organization of text, calendar date, and other data. The flexibility allowed in formatting using spreadsheet tools makes them desirable to plan, organize, and communicate schedule data. Calculations and formulas can be applied, ranges can be sorted, and data sets can be filtered using spreadsheet tools.

Schedules are easily represented using matrix representation tools such as MS Excel (Figure 9). The schedule data, presented in a list view format, are easily sorted and filtered. And the schedule entirety, which may span months or even years, can be consolidated into a succinct list. Contingency schedules, like the POA&M templates discussed in Chapter I, are easily created, applied, and reused.



B53

	A	B	C	D	E	F	G
1	Project Template: 7-Day Unit Training Deployment						
2							
3		Variable Dates:					
4		D - Deployment Date (Main)	0	(When the main body departs)			
5		A - Advance Echelon	D-5				
6		R - Retrograde	D+7				
7							
8	Task No.	Task Name	Responsible Party	Start Date	End (Due) Date	Comments	Status
9	PHASE 0	OVERALL PROJECT GUIDELINES					
10	1	Initial Planning Conference	HQ	D-30D	D-30D		
11	2	Mid-term Planning Conference	HQ	D-20D	D-20D		
12	3	Final Planning Conference	HQ	D-10D	D-10D		
13	4	Exercise Training Period	HQ	D+0D	D+7D		
14	PHASE I	PRE-DEPLOYMENT					
15	5	Submit Ammunition Request	S-3 Operations	D-30D	D-30D		
16	6	Create Personnel Rosters	S-1 Admin	D-30D	D-30D		
17	7	Submit Range Requests	S-3 Operations	D-25D	D-25D		
18	8	Submit Frequency Request	S-6 Communications	D-30D	D-30D		
19	9	Request Pay Advance	S-1 Admin	D-21D	D-21D		
20	10	Submit Airfield Support Message	S-3 Operations	D-20D	D-20D		
21	11	Submit Transportation Requests	S-4 Logistics	D-20D	D-20D		
22	12	Submit Request for Radios	S-6 Communications	D-15D	D-15D		

Figure 9. Training Deployment Schedule Represented in Microsoft Excel

3. Electronic Calendars

Calendars are a natural way for people to maintain schedules and collaborate using them. Physical calendars come in all forms and sizes. Most commonly encountered are paper-based monthly, weekly, and daily calendars. Others types of calendars include large paper, poster, and dry-erase boards, setup in a variety of calendar formats. These types of calendars are useful in conveying information to a larger audience and can aid in providing sufficient room to include many details of a schedule if needed. They allow for collaborative planning as their size can allow many people to view and discuss around the calendar. Changes are easily made on dry-erase boards and differing colors can ease in delineating information.

Electronic calendars mimic the capabilities of paper and other physical calendars. Current calendar applications allow for nearly the same type of scheduling information representation as physical calendars. Electronic calendars have the advantage of allowing differing views of the same information without having to enter the scheduling information more than once. Schedules in electronic form also have the advantage of being easily copied and replicated for collaboration, sharing, and archiving.

Computer applications such as Apple iCal and MS Outlook are commonly encountered calendar applications for desktop and laptop computers. Other examples of electronic calendars include cloud providers of e-mail such as Google and Yahoo. Mobile electronic devices also have fairly robust calendaring applications, able to maintain scheduling information synchronously with calendar host providers.

C. ELECTRONIC CALENDARING

1. History

Since the mid-1980s, electronic calendaring devices have augmented paper calendars and schedules. The appeal of electronic calendars over time has been the ease of recording, flexibility in changing, and the sharing of scheduling information. Early groupware software such as the extension to WordPerfect and Novell Groupwise made electronic calendaring appealing to desktop computer users and made collaboration of schedules between them possible. The introduction and advancement of the PDA made it possible for the electronic calendar to be portable. Web development of calendaring tools and web hosting of calendars helped to extend calendar access and manipulation outside of one’s personal desktop. Advances in mobile communication and technology have brought the seamless integration of calendar data across desktop computers and a myriad of mobile devices.

2. Current Capabilities and Features

Modern electronic calendaring systems allow individuals and groups to create, edit, view, and share calendar events with one another. These calendars “live” in the “cloud” and can be accessed from virtually any location by a computing device with a connection to the Internet. Because of the ubiquity of portable and handheld computing devices, the reliance and dependency of these electronic calendars are increasing for people both in their personal and professional lives.

Calendaring systems today, are primarily good at capturing and displaying named events—occurring at specific dates and times, with a specific duration. In addition, these events can contain other attribute information such as the event location, reminder notifications, and associate a specific calendar for each recorded event (i.e., personal, family, or work calendar). Most calendaring systems allow the ability to drag and drop (cut and paste), events for rescheduling purposes. Recurrence parameters are also commonly available allowing users to setup up recurring events such as weekly meetings, monthly due dates, or semi-annual reminders.

Users can create multiple calendars allowing organization of differing schedule information. Individual calendars can be selectively filtered on or off. This can provide a means to see schedule conflicts or gain an understanding of event interaction. Deselecting calendars can also help de-clutter the calendar. Calendars can be used on a personal basis, or may be shared with others. Varying permissions can be assigned to those that the calendar is shared with.

Calendar users also have the ability to create meeting requests or invite others to scheduled events. These invitations are often sent via e-mail to the invitees. Options to accept or reject the meeting request by the invitee can help the originator determine the number of people, and more specifically who will be attending the event. Some systems also have the capability to model organizational resources and reserve/schedule their use. Examples of the resources that might be modeled are class/conference rooms, vehicles, and support services.

3. Mainstream Calendaring System Limitations

Calendaring systems have evolved considerably in the past decade. Most of the capabilities and features discussed previously have come about in the past 10 years. These advancements have led to greater adoption and dependence of electronic calendars. Despite these advancements, calendaring systems that limit their efficiency and effectiveness, and in some cases preclude them from being used as a scheduling tool.

Aside from the deficiencies outlined in this paper, the lack of hierarchical scheduling support and lack of temporally related events, current calendaring systems also lack a robust way to manage a group of events. Mainstream calendaring systems do not allow a way to select a group of events and manipulate them together. Each event needs to be manipulated separately. This method of manipulation can be grossly inefficient, especially as the number of related events that require the common change increases.

Differing graphical and categorical representation of events on calendaring systems is also very limited using today's mainstream calendaring tools. Calendaring systems display events in the same chronological manner, not allowing differing font sizes, colors, or types to be displayed on the calendar. Images and icons can also not be used to represent graphically a calendar event. In project planning terms, event categories such as task, phase, and milestones cannot be delineated.

4. Collaboration and Ubiquity

Collaboration using electronically distributed calendars first evolved within larger business organizations with common enterprise e-mail systems. These calendars reside on the e-mail server, are associated with a user's e-mail account, and accessed through an e-mail software suite. Because the calendars reside on the server, collaboration of calendars could be achieved within the organization, but only within the scope that the e-mail system would allow. Intra-organization calendar collaboration could be achieved, but inter-organization collaboration of calendars was not natively available.

Collaboration of personal and social calendars has gained significant momentum since the introduction of web calendaring. Similar to organizational e-mail systems, the web calendar resides on a server and is accessible to the user through a web browser via a URL. Any Internet-enabled computer can access these calendars with a web browser. Additionally, smartphones and other Internet-capable portable devices can also access, modify, and maintain synchronicity with the server-based calendar. With mobile calendaring becoming more accessible and robust, users are integrating their work, family, and social calendars in one framework.

D. ICALENDAR DATA FORMAT

The iCalendar format [9] is a nonproprietary, open-source specification for the purpose of calendaring and scheduling data exchange. Because the wide variety of applications that contain some level of support for this standard, it is used in this paper as the basis for representing a generic calendaring model (henceforth referred to as the *iCalendar model* or *iCalendar data model*), agnostic of any particular implementation or application. We introduce and examine the iCalendar model in this section, and extend the model in Chapter IV to achieve hierarchical calendar and temporal event capability.

1. Background

The Internet Engineering Task Force (IETF) specification, RFC 5545, is MIME content based and can be delivered using a number of protocols including, but not limited to, SMTP and HTTP. It superseded an earlier version, IETF RFC 2445, in September 2009. RFC 2445 was based on the original attempt at open source modeling of calendaring and scheduling information by the Internet Mail Consortium's (IMC) vCalendar format. It was renamed to iCalendar in RFC 2445 to avoid confusion.

The IMC (http://www.imc.org/pdi/) initiative in the mid-1990s sought cross-platform interoperability of calendaring and scheduling data. It aimed to allow people using different platforms and vendor specific calendaring software to send and receive calendar and schedule information using different transports across the Internet. Conceptually, across differing product domains, it would allow scheduling of meetings using e-mail, distribution of project or event schedules, and web calendar publishing. Originally derived from Versit's (a consortium of industry leaders made up of Apple, AT&T, Siemens, Lucent Technologies, and IBM) Personal Data Interchange (PDI) technology, vCalendar was transferred to IMC for greater promotion and adoption by industry.

Related specifications used in conjunction with iCalendar are iTIP [10] and iMIP [11]. ITIP, RFC 2446 and later RFC 5546, defines how iCalendar is used to support group scheduling (i.e., inviting users to a meeting and receiving their replies). IMIP, RFC 2447 and later RFC 6047, is the specification that allows iTIP scheduling in conjunction with e-mail.

iCalendar is a text-based format with its own vocabulary and grammar. Data files typically have an .ics file name extension. An equivalent XML-based specification- xCal [12] - provides a one-to-one mapping of iCalendar definitions and constructs to xCal.

2. Data Model Description

a. Core Object

The iCalendar format consists of a top-level element called the Core Object, a collection of calendaring and scheduling information. This top-level element typically contains a single iCalendar object (calendar), but may consist of multiple iCalendar objects. Each object contains a list of calendar properties and one or more calendar components. A simple example of an iCalendar object is given below [9]:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
UID:19970610T172345Z-AF23B2@example.com
DTSTAMP:19970610T172345Z
DTSTART:19970714T170000Z
DTEND:19970715T040000Z
SUMMARY:Bastille Day Party
END:VEVENT
END:VCALENDAR
```

b. Calendar Components

Each object (calendar) can contain one or more iCalendar components. These components can help describe properties of the object (e.g., TIME ZONE component), are scheduling information (e.g., EVENT component), or augment capabilities of the schedule component (i.e., an ALARM component associated with an Event component).

The basic components most frequently encountered by people while using calendaring tools are the Event and TO DO components. The Event component is a scheduled event consisting of a specified start or end date and time. A TO DO component is a task with no necessary start or end date and time. Figure 10 shows the iCalendar core object, object, and calendar component structure.

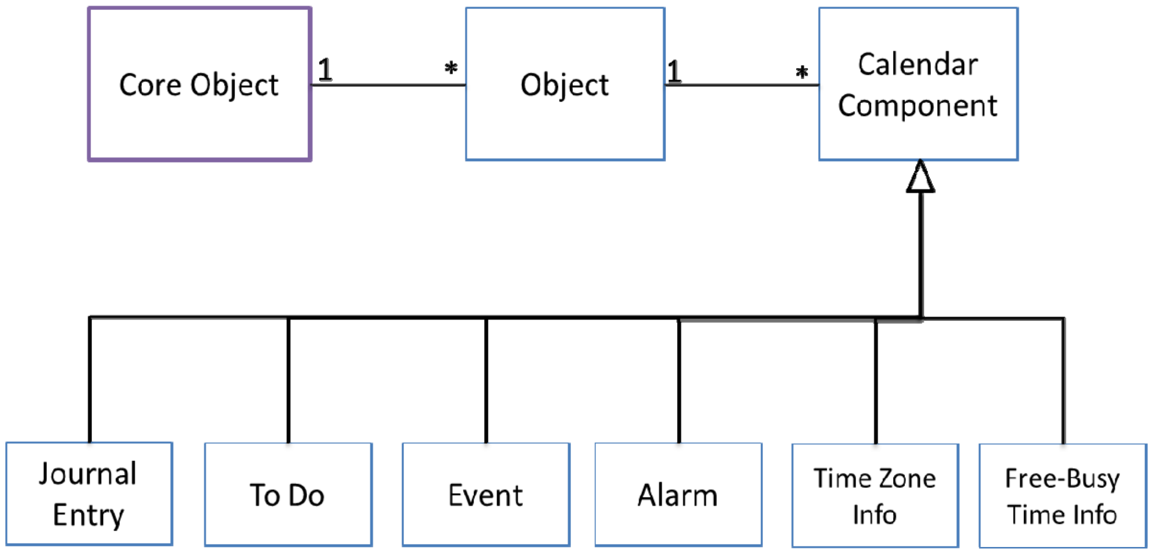


Figure 10. Structure of iCalendar Data Model

c. *Properties and Value Data Types*

Various properties apply to objects and calendar components. For example, the property *PRODID* (product ID) is specific to describing the globally unique identifier of the vendor for which the calendar object was created and is applicable only to the calendar object. The value type is text. The *DUE* property is applicable only to a TO DO component and can be specified only once. It has a *DATE-TIME* value type which specifies a specific date and time (e.g., 19980118T230000—representing 2300 on 18 Jan 1998). Both of the previous properties could only occur once for the referenced object or component. The iCalendar specification provides the semantic details for what value types are associated for each property, and associates properties with calendar objects and components.

d. iCalendar Extensions

The iCalendar specification provides a framework for use and inclusion of nonstandard properties. Nonstandard properties are named with the following prefix “X-” and are encountered quite frequently when examining ics-files. No property exists for the calendar object name, thus the nonstandard property *X-WR-CALNAME* is a fairly standard way to describe the name of a calendar.

3. Adoption

iCalendar is the interoperable standard for calendar data interchange [13]. It is used to interchange calendar information across mobile, desktop, and other devices. Major calendar application providers all use it as a means to import/export calendar data, both automatically through calendar synchronization, and manually through file exchange.

Besides calendar software, the iCalendar standard has been used as a basis for the web service-scheduling standard, WS Calendar 1.0 [14]. Service Oriented Architecture (SOA) and Web Services is an increasingly growing framework for business automation. SOA allows for a system of distributed solutions based on Web technologies on a global scale. It achieves this through a universal model for representing logic and information that is globally accepted.

SOA and Web services are also increasingly being used to automate and coordinate physical services [14]. As a result, many semantic standards and protocols have been established for use in industrial, home, and energy automation services. Such examples include:

- <u>Open Building Information Exchange (oBIX)</u>–Enabling mechanical and electrical control systems in buildings to communicate with enterprise applications

- Emergency Management–Enabling information exchange to advance incident preparedness and response to emergency situations
- Energy Interoperation–Enabling the Collaborative and Transactive use of Energy
- Web Services for Building Automation and Control Networks (BACNet/WS)–Industrial and home automation standard
- OLE (Object Linking and Embedding) for Process Control Unified Architecture (OPC-UA)–Industrial automation standard

Central to these types of physical services is the notion of scheduling. As this domain of service grows, so too does the interaction and interdependencies amongst the different service components. A common scheduling standard is imperative to ensure their seamless integration and automation. WS-Calendar is the web service extension that fulfills that.

III. REQUIREMENTS

A. ASSUMPTIONS

The motivation for this work is based on the premise that the electronic calendar is an individual's primary computer-related scheduling tool of choice. The electronic calendar gives every user the ability to easily share, add, selectively view, and overlay schedules that no other tool can. It can exist seamlessly across multiple devices, and in some cases at little to no cost to the user. For these reasons, it is believed that adoption of a separate scheduling tool, which allows for time-dependent event relationships and hierarchical schedules, will not take hold for the masses.

Indeed, the project planning software discussed earlier includes such capabilities; due to cost and complexity, however, they remain outside mainstream use. A more planetary approach is to add these capabilities to calendaring systems and allow ordinary users more capability to perform project-planning type schedules using their primary scheduling tool. By extending the capabilities of current calendaring systems to allow for more robust scheduling capabilities we better satisfy the way the military (and other organizations and institutions) develops and publishes exercise and contingency planning schedules. Instead of an inefficient and ad hoc way of using and merging spreadsheet scheduling information with electronic calendars, the set of extensions proposed would allow military planners to build template schedules, use/reuse them as necessary, and have them be represented in a variety of formats for publication and dissemination.

The following specification relies on the basic calendaring functionality and capabilities described in Chapter II. It also relies on the iCalendar data model as basis for implementation. Use cases and features for a generic calendaring system are specified at the CalConnect website [15]. The requirements set forth are to be considered extensions to the use cases that describe the requirements in the iCalendar model.

B. SYSTEM OVERVIEW

The following requirements specification follows the Unified Process [16] approach for software requirements specification, categorization, and documentation. It is not meant to be an exhaustive or complete specification. Instead, we use it to help illustrate and envision how such a calendaring system with the capabilities advocated would be used and further reveal the motivation for why they are needed.

1. Stakeholders

a. Organizations

Planning and scheduling are key tasks for any organization. Tools that allow for increased integration and visualization of integrable schedules will help increase productivity and reduce misconceptions and errors. Many organizations have made investments in corporate e-mail and calendaring software such as MS Outlook. Other scalable solutions exist, as well, such as Google Apps for Business. Enhancements to these calendaring tools will allow organizations to conduct more realistic large-scale project-planning type efforts. Organizations (and even individuals) wishing to create and re-use project schedule templates will be able to do so efficiently and quickly.

b. Managers

Managers are key decision makers and resource allocators within a project framework. Tasks within the manager's area of responsibility are done with higher-level project constraints and decision points in mind. The manager's framework and structure for task accomplishment is subordinate to the overall project framework, and should be modeled as such in a project schedule. These proposed calendaring tools will give managers such integration.

c. Individuals

Whether the project manager, active participant, or outside contractor; all individuals are stakeholders in the overall project. Changes that are made in the overarching timeline will have ripple effects to all the participants involved. These changes can be quickly made to the schedule. Because events can be related temporally, ripple effects can be made automatically, quickly, and accurately.

2. Actors

a. Creator

This user will be the schedule owner. The *creator* will have overall authority over that particular schedule and any relevant subschedules. The *creator* will also specify permissions, policy enforcement, and delegation of rights. Also, other capabilities that are discussed later will be part of the *creator's* purview, such as schedule setting the schedule window or a schedule freeze. Note that it is possible to have more than one creator for a particular schedule. The *creator* can also create component schedules for subordinate organizational groups.

b. Editor

Editors will have most of the functionality that *creators* have. They can set selected permissions and policies, and grant access to other users (up to *editor* only). *Editors*, however, cannot grant *creator* access control and cannot set permissions and policies in contradiction to what the *creator* has set. *Editors* can create subordinate schedules, and both the *creators* of the existing schedule and the *editor* who created the subordinate schedule become *creators* of the newly created schedule. The *editor* permission will normally be given to subordinate managers or their delegates.

c. Viewer

A user who has been given permission to see the schedule only is a *viewer*. *Viewers* have no ability to create or modify schedules or events. They can however see subordinate schedules for which the user has been granted *viewer* permission but cannot see the schedule of its parent. Project participants who do not have schedule authority will typically fall in the *viewer* actor category.

d. System

The System is the calendar application and its data store. The system can be an enterprise-type calendar software application or can also be a stand-alone application on a mobile device or computer. The system will be able to read and interpret existing calendar data [9], as well as the extensions proposed in Chapter IV.

3. Functional Requirements (Features)

The software features herein are necessary to provide the type of functionality desired in a system that provides the capabilities that are sought. A list of these essential features can be found in Table 1.

Table 1. Features Desired in Calendaring Systems

Feature ID	Feature Name	Description
1.0	**Display/View**	
1.1	Toggle Selection of Visual Relationships	Ability for user to selectively turn off/on the display of temporal relationships. With this feature turned off, the view will be less cluttered.
1.2	Gantt Chart Representation of Project	Gantt representation is not currently a view option for most mainstream calendars. This is a vital viewing format for project management.
1.3	Visual Distinction Between Child/Parent/Static Events	It is important for users to quickly and easily distinguish these types of event objects when displayed on a calendar.
1.4	Selective Display of Calendars	Ability to show differing schedules in the calendar hierarchy is needed. Show selections will automatically cascade down the hierarchy (i.e., when someone de-selects a calendar, that de-selection will apply to all the children). However user has selective control of any schedule combination desired.
2.0	**Permissions / Access Control**	
2.1	Schedule Lock	Ability to selectively prevent schedule changes.
2.2	Version Control	Ability to have version control over the project schedule.
2.3	Project Window Constraining	Ability to set overarching time windows for project schedule (i.e., constrain users from scheduling events outside of the project window).

4. Non-Functional Requirements

Interoperability is a key feature of calendaring systems today. An event can be saved and sent as an attachment to an e-mail and accepted into a receiving user's calendar. A calendar created in one application can be easily added to a differing application. Any new calendaring system must also remain interoperable. Interoperability is facilitated by standards (in our case [9]). Since we are discussing new functional capabilities for calendaring systems, these must translate and work their way back to the standard for interoperability to occur at this new level. In Chapter IV, we discuss how the iCalendar format [9] can be extended to facilitate these new capabilities. Any long-term use and pseudo adoption of these extensions must eventually work their way back to the standard to be incorporated for long-term durable success.

Even being implemented as part of the standard vocabulary, manufacturers of calendaring systems will adopt these new capabilities at different times or not at all. A standard way to handle the migration of information from an application using time dependent event relationships and hierarchical calendars to another that does not must be allowed for. To not allow for it at all would be undesirable. Instead, it is more probable to conceive of the application converting time dependent events to static events and hierarchical schedules to individual calendars.

Interoperability between other sources or representations of calendaring data is also desired. This is particularly useful for template schedules. As discussed in the introduction, military planners use the POA&M format for developing and maintaining schedules for operations and exercises. Although there is no single POA&M format, arguably all contain the necessary information to be replicated in a calendaring system with these new capabilities. Doing so in a quick and seamless way is also a requirement.

Calendar applications must continue to be usable by the mainstream masses of users. It is doubtful that the new calendaring system tools will be used by any great majority of all users therefore the new capabilities must not distract or confuse those who have no intent to use them. A separate mode of operation is not desired but to some degree this is probably a bit realistic. For ordinary calendars and events, these new capabilities will not be visible or accessible. Only when working within the defined scope of the project schedule will these features be apparent and useful.

C. USE CASES

Use cases are text stories used to describe the prescribed system and record the requirements expected of such a system. The method for use case development and documentation is followed from Larman [16]. Table 2 lists the use cases considered unique and essential for a calendaring system that features time dependent events and hierarchical calendars. Note the primary actor listed in Table 2 is the lowest actor listed (e.g., Viewer implies *editor* and *owner* are also actors). A complete description of all use cases can be found in the Appendix.

Table 2. Calendaring System Use Cases

Use Case ID	Use Case Name	Primary Actor	Complexity
1.0	**Creation and Modification of Temporal Events**		
1.1	Create Relational Event	Editor	Low
1.2	Modify Static Event to Relational Event	Editor	Low
1.3	Modify Relational Event to Static Event	Editor	Low
1.4	Modify Temporal Relationship Between Parent/Child Events	Editor	Medium
1.5	Modify Parent Time Properties (Project Timeline Shift)	Owner	High
1.6	Delete Temporal Event (Child)	Editor	Low
1.7	Delete Temporal Event (Parent)	Editor	Medium
2.0	**Visual Requirements**		
2.1	Toggle Selection of Visual Relationships	Editor	Medium
2.2	Gantt Chart Representation of Project	Viewer	Low
2.3	Matrix View of Project	Viewer	Low
2.4	Selective Viewing of Schedules in Hierarchy	Viewer	Low
3.0	**Schedule Hierarchy**		
3.1	Create New Child Schedule	Editor	Medium
3.2	Move Schedule Within the Hierarchy	Owner	Medium
3.3	Freeze Schedule	Owner	Medium
3.4	Delete Schedule (Project Cancellation)	Owner	High
3.5	Save Schedule (Project Archival)	Editor	High
3.6	Reuse Archived or Existing Schedule	Editor	High
3.7	Publish (Share) / Subscribe Schedule	Editor	Medium

Out of the many use cases presented in Table 2, we focus on three use cases that highlight the benefits of the proposed capabilities outlined in this paper. These use cases show the utility for relating events temporally for projects as well as illustrates why structuring schedules hierarchically makes sense.

1. UC 1.5–Modify Parent Time Properties (Project Timeline Shift)

Changes in project timelines occur all too often, for a variety of reasons. They are moved ahead of their original timeline because of urgency or necessity, or they are delayed because of resource constraints. Projects that revolve around a critical event, such as the day of airlift for a military unit's pre-deployment preparation schedule, can have repercussions throughout the schedule if that critical event is changed.

When the entire project timeline is moved-up or delayed by a certain time period, and the internal structure of the project is to be kept intact, it would be desirable to be able to adjust the critical event and have the rest of the project adjust accordingly. This is especially true because today's calendaring tools make such a shift in timeline cumbersome and error prone. Instead of just modifying the critical event, each individual event is required to be changed. For some small projects occurring in a relatively tight timeframe, the total amount of project restructuring may not be too difficult. But for larger projects that span across months to years, the amount of restructuring can be difficult, tedious, and prone to error with each adjustment.

2. UC 3.3–Freeze Schedule

For projects that involve complex organizational planning and interaction, the ability to deny changes to the schedule(s) may be an indispensable feature. Change management [17] requires some degree of control by the project manager and its participants. Large projects—with many suborganizations, events, and planners—can have unintended and unknown repercussions when even the smallest change is made to a schedule. For these large projects, managers need tools to enforce change or to prevent changes.

Freezing the schedule is one method to prevent changes. A schedule freeze, initiated by the owner of the schedule, would prevent changes from occurring until either the freeze is lifted, or unless a particular change is approved (by the owner). The freeze can either cascade throughout the schedule structure or could be imposed on a single schedule.

The application of the schedule freeze to a project schedule may be initiated for a variety of reasons. One such reason is to initiate a review of a schedule for quality assurance or management approval. Another reason for a schedule freeze may be due to the process methodology for a project. Organizations that follow a waterfall-type process, where project planning may be a deliberate phase in the overall project execution, may rely on the ability to freeze the schedule. This is not to say that further schedule changes cannot or should not be made, but rather that enforcement of a change management process is necessary to effect the change. These types of processes are designed to reduce or eliminate the unintended consequences that happen when schedule changes are made

3. UC 3.6–Reuse Archived or Existing Schedule

Once a schedule has been developed and created, it will likely be used as a model for some future reoccurrence of that event. This is especially true for military planning. Exercise and contingency planners generally follow the same approach for developing exercise and contingency plans—obtain what has been done previously (e.g., previous planning templates), analyze and reuse those portions that are relevant (updating them with current information). For the schedules associated with these plans, often the same milestones, phases, and tasks will be used to make and track progress.

Calendaring tools that captured the type of information in Figure 2 (POA&M with non-specified dates) would allow users to rapidly create schedules with relevant dates and times by giving specific date and time information to those variables left unspecified in the template schedule. The template schedule would need to contain the appropriate temporal event relationships (Chapter IV) necessary to produce the schedule structure.

IV. DESIGN

A. TEMPORAL RELATION CONCEPTS

In order to achieve the desired functionalities and capabilities described in Chapter III, the following concepts are presented. The type of temporal dependency, the duration of the event, and the time offset between events are necessary to establish the dependency between events. Inheritance properties are necessary to establish relationships between the independent and dependent events. It is also a necessary concept for the establishment of calendar hierarchies.

Project scheduling can be difficult and complex. Physical, technological, and resource constraints affect the overall project [18]. The project schedule is directly affected by time and resource available. The scheduling of an event may be based on necessary preconditions or post conditions that must adhered to. In this thesis, we limit our study to modeling deterministic time constraints.

In a new-home landscape project (Figure 11), for example, the sprinkler pipes and heads must be installed sometime after the trenches have been dug. After the sprinklers have been installed, the soil can be watered; then one day (24-hours) later, the sod can be put down. The first two sequences have no specific interval between events, only that they are performed in sequence (with the start of one not occurring until after the completion of the other).

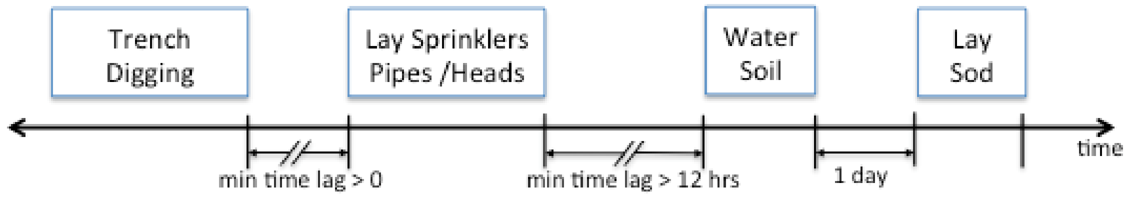


Figure 11. Landscape Scenario with Nonspecific Time Intervals

[19] refers to the interval between such events as the minimum time lag. For this example, the minimum time lag is zero, meaning the events can occur one after the other

with no lag between them. This is not necessarily always the case, however. If, for example, the epoxy used to seal and adhere the sprinkler pipes required a 12-hour cure time, the minimum time lag would be 12 hours and the watering of the soil could begin no earlier than 12 hours after the sprinklers were installed. These types of interval relationships are nonspecific and cannot be modeled as such using calendaring tools. Although potentially of use for automated or computer assisted scheduling optimization, we avoid any ambiguous dependencies because of their complexity and diminished benefits in real-world project scheduling.

We instead focus on fixed intervals, such as the time interval between the watering of the soil and the laying of the sod. In Figure 11, this interval is specified to exactly one day (i.e., one day after the soil has finished being watered, the sod can be put in). These types of fixed interval relationships can conceivably be modeled using mainstream calendaring tools. Whereas the non-specified interval constraints could not be easily modeled using calendaring tools, the application of fixed time intervals to uncertain time intervals would be more indicative of an actual project schedule and could be modeled using calendaring tools. The landscape scenario with specific time intervals is shown in Figure 12. This scenario can be implemented in mainstream calendaring tools.

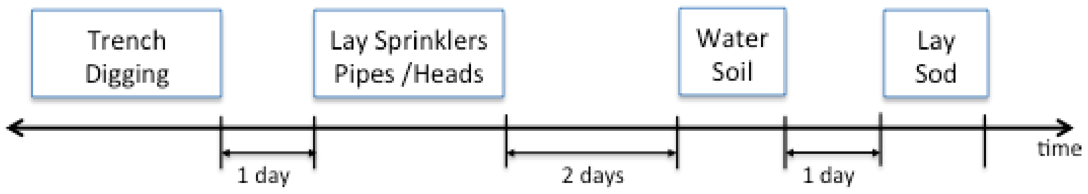


Figure 12. Landscape Scenario with Specified Time Intervals

Of interest, therefore, are those time-dependent event relationships that are unambiguously defined and can be modeled and displayed in a calendaring tool. Understanding that a scheduled event can be displayed on a calendar given two out of the three characteristics (start time, duration, or end time), calendar events can be categorized as follows:

1. The start time and the end time are known.
2. The event duration and either the start or end time is known.
3. The event duration is known and the start or end time can be calculated based on an interval from another event's start or end time.

Categories 1 and 2 exist in current calendaring tools. Category 3 is a time-dependent relationship between two events. The independent event has start and end properties (specified or derived) and the dependent event's start and end times can be derived from the event's duration and interval relationship with the independent event. It is important to note that the label of independent and dependent event has scope only in reference to the binary relationship between two relatable events. Within the scope of Events 2 and 3 (Figure 13), for example, Event 2 is the independent event and Event 3 is the dependent event. Outside of this binary relationship, these identifying labels do not necessarily hold. For example, between Events 1 and 2, Event 2 is the dependent event and between Events 3 and 4, Event 3 is the independent event.

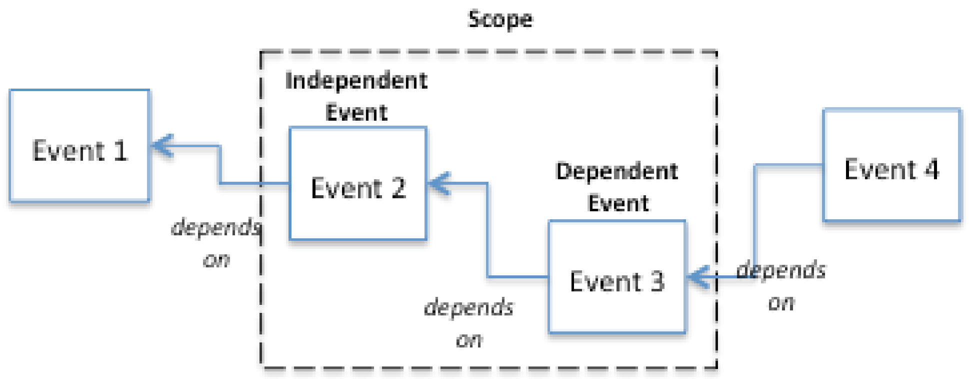


Figure 13. Scope of Independent and Dependent Labeled Events

Exploring Category 3 type events further, it can be seen that four interval relationships can be specified between two pairs of relatable events—either the dependent event's start time is determined from the independent event's start (SS) or end (SE) time, or the dependent event's end time is determined from the independent event's start (ES) or end (EE) time. The interval relationships: SS, SE, ES, and EE describe the temporal relationship between the dependent event and the independent event. The first letter of the interval relationship describes the dependent event's (DE) related time property (either S–start or E–end), and the second letter describes the independent event's (IE) related time property. The second part of the interval relationship is the offset value. This value is specified as some amount of time, either positive or negative. A positive offset value refers to a time interval in the future, whereas a negative offset value refers to a time interval in the past (from the intended reference point). Each of the four relationship possibilities is discussed in more detail below.

1. SS–Start of Event Depends on Start of Independent Event

Figure 14 describes the SS interval relationship. This type of interval relationship is one of the more frequently used and encountered in normal scheduling. The illustration in Figure 14.a says, "the dependent event starts sometime after the independent event starts." This is achieved with a SS interval relationship and a positive offset value. As an example, this type of time constraint would be useful for staggering departure times for a military lift movement involving multiple waves of aircraft. The illustration in Figure 14.b shows how the SS interval relationship can be used with a negative offset. This says to the effect, "the dependent event starts sometime before the independent event starts."

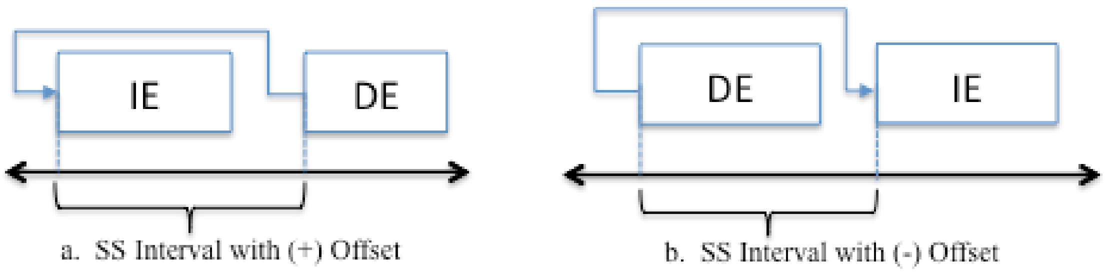

IE
DE
a. SS Interval with (+) Offset
DE
IE
b. SS Interval with (-) Offset

Figure 14. Start to Start (SS) Interval Relationship

Although Figure 14 shows the two events occurring one after the other, it does not mean to imply that the events cannot occur concurrently. Figure 15 shows the SS interval relationship with events overlapping.

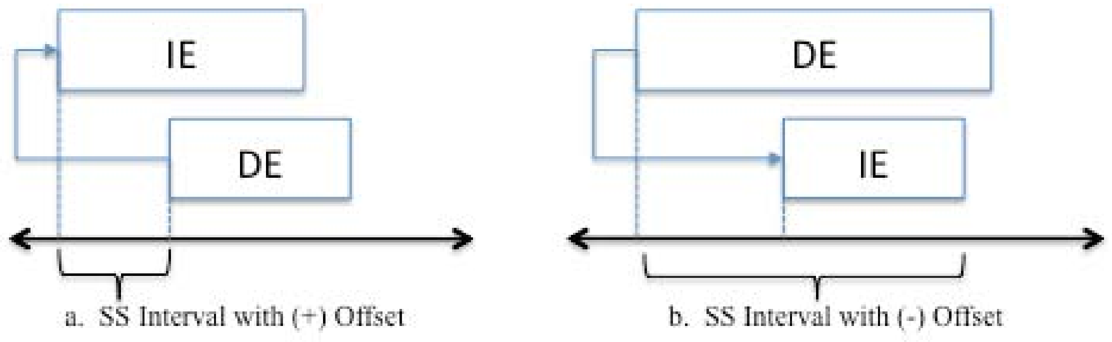
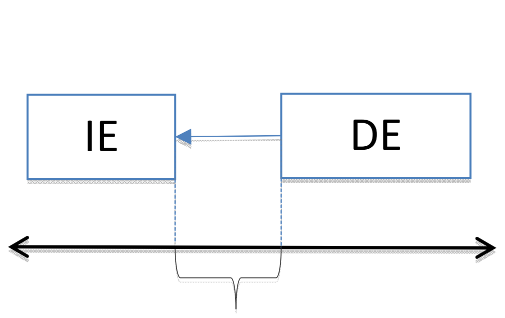

IE
DE
a. SS Interval with (+) Offset
DE
IE
b. SS Interval with (-) Offset

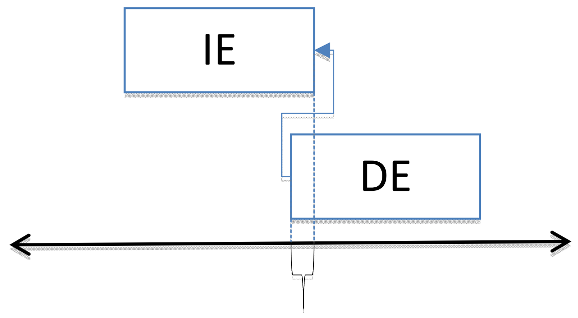
Figure 15. Start to Start (SS) Interval Relationship with Overlapping Events

2. SE–Start of Event Depends on End of Independent Event

Figure 16 describes the SE interval relationship. The illustration in Figure 16.a is fairly intuitive and occurs frequently in describing events in series. It says, "the dependent event starts sometime after the independent event ends." This is achieved with a SE interval relationship and a positive offset value. An example of this type of time constraint would be useful for round robin scheduling of a single aircraft travelling to multiple destinations. The independent event is the first leg of the trip, the interval is the time on deck at the airport, and the dependent event is the subsequent leg. The illustration in Figure 16.b says to the effect, "the dependent event starts sometime before the independent event ends." Continuing on the aviation example, this type of interval relationship would be useful to schedule a flight departure 30 minutes before another flight's arrival to prevent both aircraft being at the same gate at the same time.



a. SE Interval with (+) Offset

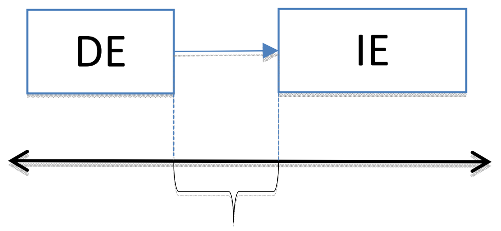


b. SE Interval with (-) Offset

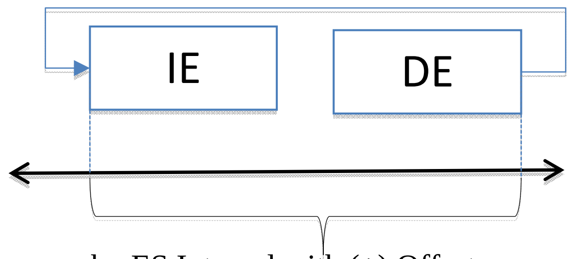
Figure 16. Start to End (SE) Interval Relationship

3. ES–End of Event Depends on Start of Independent Event

Figure 17 describes the ES interval relationship. The relationship is essentially that described by the SE relationship but with the dependencies reversed. For example, instead of specifying "the dependent event starts sometime after the independent event ends," the same can be achieved by saying "the independent event starts sometime after the dependent event ends" (Figure 17.a). The effect is the same, but the role of independent and dependent event is reversed in the SE to ES relationship. The same is true for Figure 17.b.



a. ES Interval with (-) Offset



b. ES Interval with (+) Offset

Figure 17. End to Start (ES) Interval Relationship

4. EE–End of Event Depends on End of Independent Event

Figure 18 describes the EE interval relationship. This relationship is similar to the SS interval relationship but the time properties for both events are specified in terms of the end vice start. Figure 18.a essentially says “the dependent event must end some time after the independent event ends.” This is achieved with an EE interval relationship and a positive offset value. As an example, this type of time constraint would be useful for staggering arrival times for a military lift movement involving multiple waves of aircraft. Figure 18.b says to the effect, “the dependent event ends sometime before the independent event ends” and can be achieved with an EE interval relationship with a negative offset.

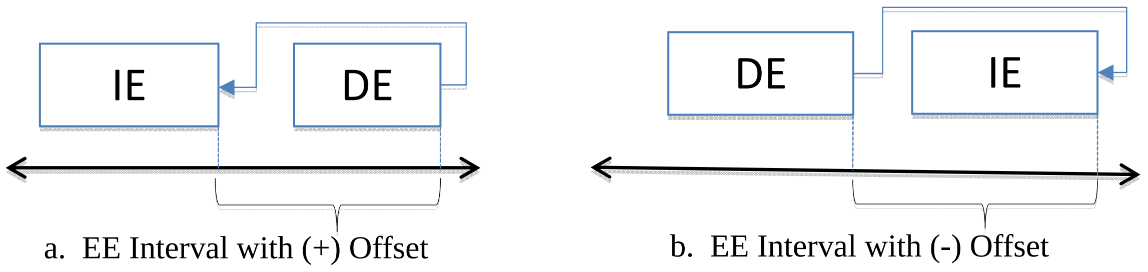

IE
DE
a. EE Interval with (+) Offset
DE
IE
b. EE Interval with (-) Offset

Figure 18. End to End (EE) Interval Relationship

B. DOMAIN MODEL

With respect to delineating schedules into individual calendars and scheduling discrete events, the domain model shown in Figure 19 can describe today’s mainstream calendaring software approach. The model is described using Unified Modeling Language (UML) notation [16]. A person organizes schedule information in separate calendars. These calendars are standalone or shared. If shared, these calendars can appear to be native to a user in calendar application, allowing them to modify events and their properties seamlessly depending on the permission level granted. EVENTS and TASKS (TO DO) are contained in their respective calendars. This application, whether it is an application on a PC, mobile device, or web app displays the calendar(s) and the events and tasks belonging to that calendar.

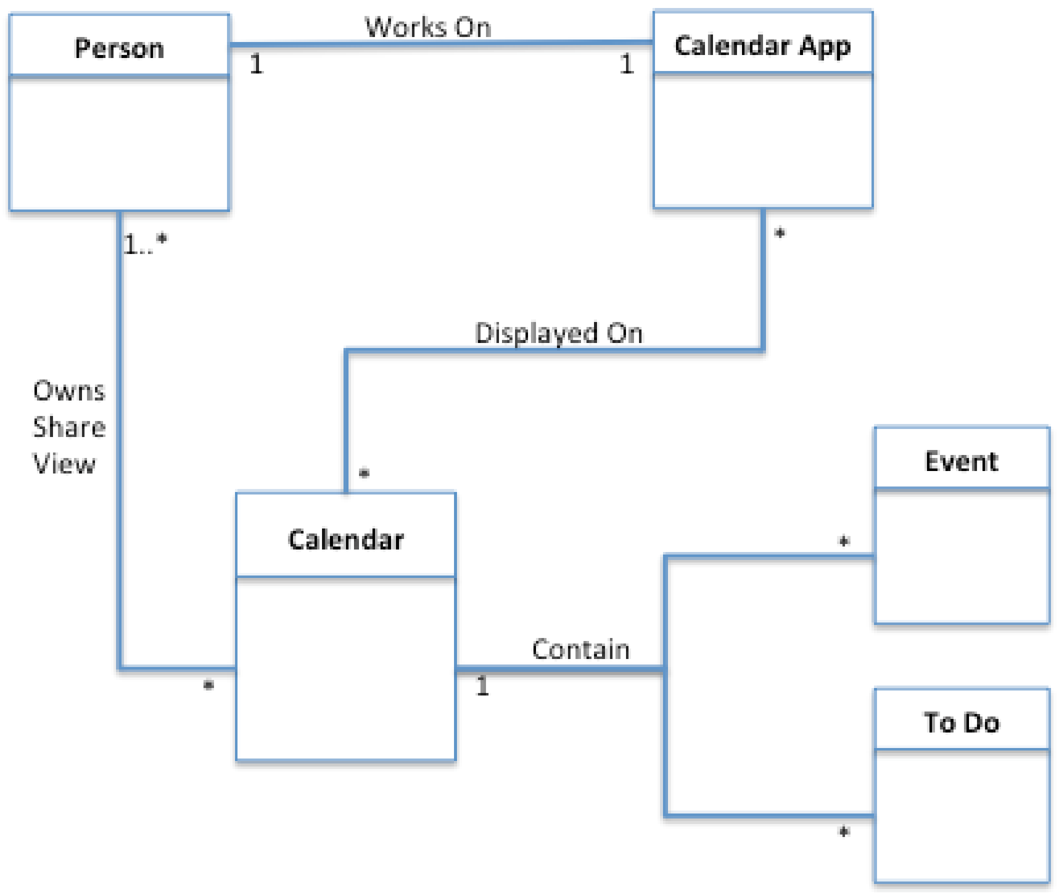


Figure 19. Current Calendaring Domain Model

The model shown above is simplified, but it captures enough details that are relevant to the topics studied in this thesis. For example, free/busy scheduling is a feature that is found on many calendaring systems. This feature helps to identify available time among a group of users and resources in order to schedule a meeting among the required participants. We leave this and other aspects of modern calendaring software out of the model for simplicity. We only concern ourselves with those aspects related to managing project schedules within a hierarchical organization and relatable time dependent events.

The domain model shown in Figure 19 satisfies the needs of individual scheduling and to some degree the collaborative scheduling needs among individuals. For organizational planning however, this simplified model has limitations. Since in this model, only users are represented, a user account must then represent that organization. The members of that organization and the suborganizations beneath it cannot adequately be represented in this model. Time-series events and other relatable characteristics between events in different calendars are not accounted for in this model.

The domain model shown in Figure 20 attempts to more accurately illustrate the situation for which organizations manage projects and project schedules. It contrasts the domain model in Figure 19 in several ways. First, the term *schedule* is used as the container of events instead of *calendar*. In some respects, these two terms are fairly synonymous, but the characterization of schedule is more appropriate for a group of relatable tasks and events, especially with respect to project management. The distinction between schedule and calendar is also useful for implementation purposes because the calendar in terms of [9] specifies the situation and condition for the current calendaring implementation paradigm. The model presented in Figure 20, and the extensions proposed in this chapter, are meant to augment rather than replace current calendaring practices.

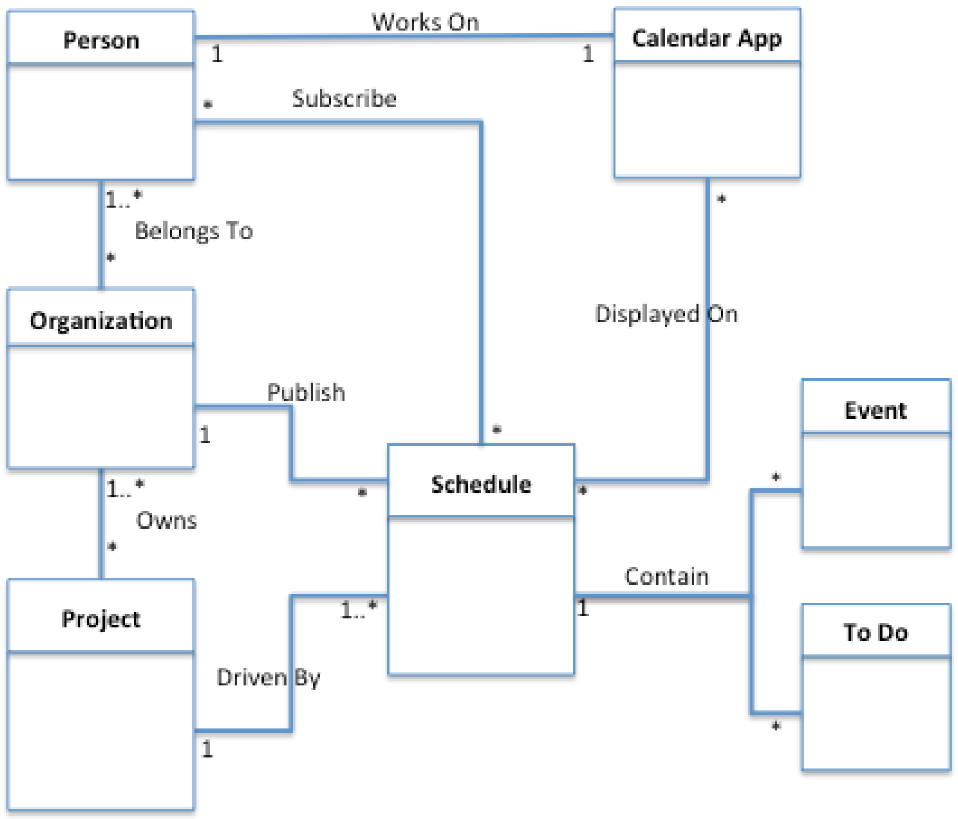


Figure 20. Organizational Scheduling Domain Model

The other key differences are the introduction of organization and project in the domain model. In the previous model, organization and project can be implied based on who the person is and what name they give the calendar. For example, if the person is the project manager for a project called CAL, and a calendar is created called CAL Master Calendar, the organization and project name can be implied from the calendar title.

Another key concept is the idea that the project and schedule belong to the organization instead of the person. Users will inevitably create and modify the schedule, but the organization retains ownership. This does not preclude a single user from following this model and utilizing the proposed features. In that case, the user and organization would be synonymous.

Lastly, the publish and subscribe concept is used to disseminate the project schedule to those needing it vice the idea of sharing a calendar. The traditional notion of sharing a calendar on an enterprise calendaring system extended to other users on that system only. Although some applications (Google Calendar) still call it “share,” the concept being used is publish and subscribe. This concept serves to share the calendar with any users with an e-mail address, regardless of whether they are in or outside of your network. Semantically, publishing a schedule has a more authoritative and formal connotation than sharing a schedule. This is desirable considering the project schedule may have a number of participants, both internal and external of the project owner.

C. DATA MODEL

To achieve the capabilities inferred by the requirements specified in Chapter III (hierarchically-related calendars and time dependent events), new calendar types and calendar components need to augment the existing data model. Currently, the relationship between calendars and events (with respect to [9]) can be viewed as depicted in Figure 21. As illustrated, a calendar is composed of many events. Specifically, the calendar can have zero or more event objects. Although calendars can contain other components (e.g., TO DO, alarms, journals), we show only the relationship between the CALENDAR and the EVENT component to focus the discussion.

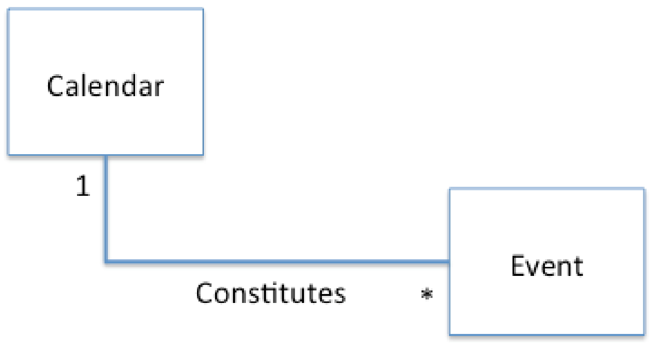


Figure 21. Current Calendar Data Model

Figure 22 shows a more robust approach to modeling subdivided schedules and time dependent events. As discussed earlier, we use the concept of a SCHEDULE instead of the CALENDAR as in Figure 16. Like the CALENDAR, the SCHEDULE object is composed of EVENTS, but also may contain zero or many SCHEDULES. This class self-reference gives the ability to create a hierarchical schedule structure that can mimic an organization's structure.

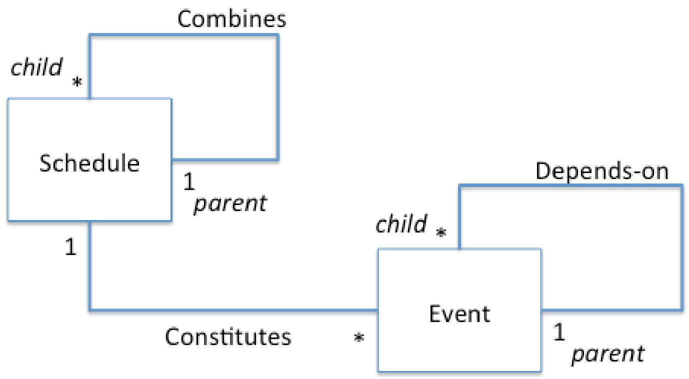


Figure 22. Proposed Calendar Data Model

The same self-referencing relation is applied to the EVENT class. EVENTS that depend on the start or end of another event will have a two-way reference to that object and will specify the type of dependent relation (i.e., SS, SE, EE, or ES) and the offset time.

The more complete iCalendar data model is shown in Figure 23. The model shows the calendar application consisting of an aggregation of calendars. Each calendar is made up of an aggregation of calendar components: JOURNAL, TO DO, EVENT, ALARM, TIME ZONE, and FREE BUSY. Figure 24 illustrates the extensions to the iCalendar data model. In object-oriented terms, the schedule is a child class of the calendar, consisting of the same components as the schedule, but also containing a new component—the relational event.

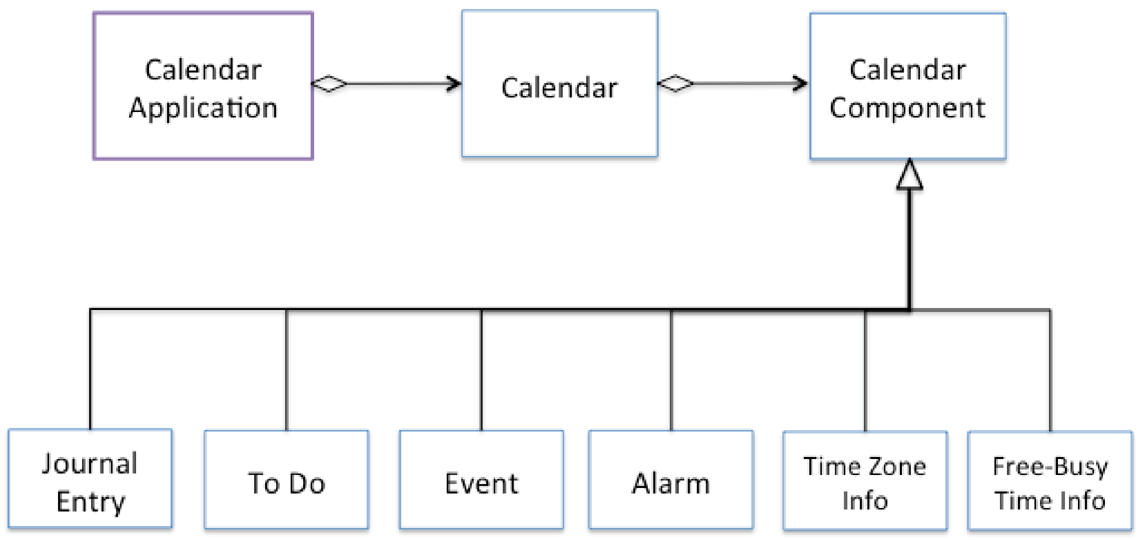


Figure 23. iCalendar Calendar Data Model

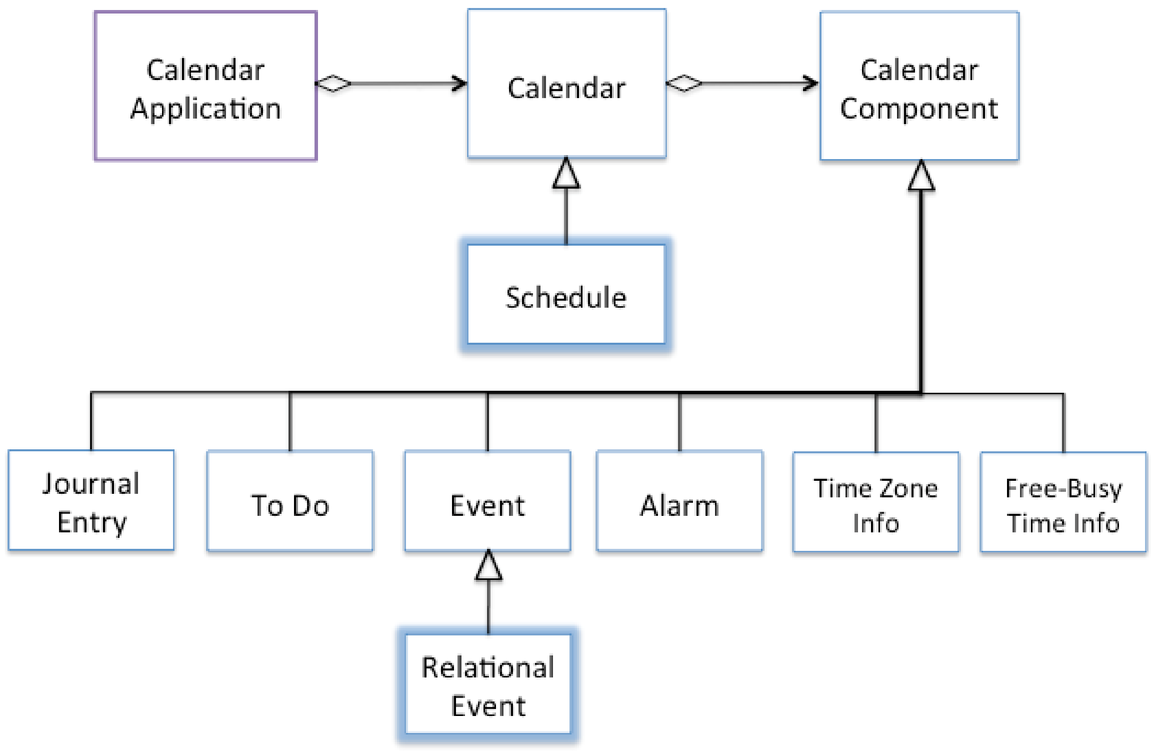


Figure 24. iCalendar Data Model with Extended Classes

With a more robust scheduling data model, the type of scheduling relationships inherent in a multi-tiered organization can be realized. Figure 25 gives a generic example of how separate but related schedules may have interacting event dependencies within their respective schedules. In this example, the objects that are shaded (Schedule 1 and Events 1.0, 1.3, 2.a, and 3.a) are parent objects with children. Events 1.3, 2.a, and 3.a have parents themselves as well as children.

Schedule 1 has two component calendars, Schedules 2 and 3. Not only is there a relationship established between the parent and child schedules, but also Schedule 1 properties can be inherited by its children. Although not shown, Schedules 2 and 3 could also have subschedules.

The types of event relationships are shown in the adjoining lines connecting the events. Event 1.0 spans across the majority of the timeline and is illustrative of a project with overall project start and end times. Events 1.1 and 1.3 show the potential for pre-start and post-ending actions. The ES relationship between Events 1.1 and 1.0 indicates the end (E) of Event 1.1 must be complete before the start (S) of Event 1.0. Although not shown, an offset would be specified to determine the amount of time required between the end of Event 1.1 and the start of Event 1.0. In this case, the offset would be 0 indicating that Event 1.1 would be scheduled to complete right when Event 1.0 begins.

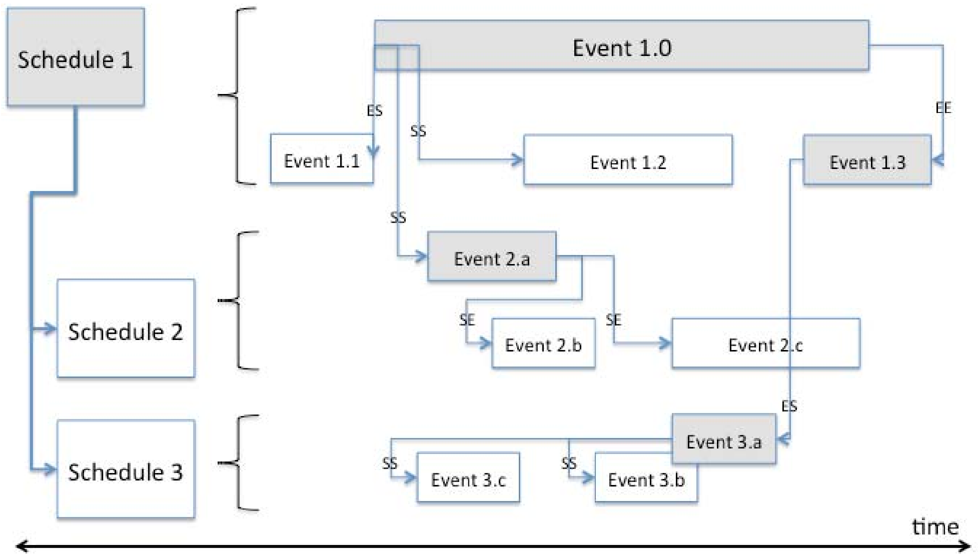


Figure 25. Hierarchical Calendars and Time Dependent Events

Dependent events can link between schedules. Event 2.a is part of Schedule 2 but its time dependency is with Event 1.0 in Schedule 1. Similarly, Event 3.a in Schedule 3 is dependent on Event 1.3 in Schedule 1. Event 1.3 also illustrates that dependent events can also have dependencies themselves. Event 1.3 depends on the end of Event 1.0 and similarly Event 3.a depends on the start of Event 1.3. The illustration in Figure 25 can be depicted in matrix format. Table 3 shows the dependency table for the example. Note, for those dependent events that take place before the start or end of the parent event, a negative offset value is used. Events 2.b, 3.a, 3.b, and 3.c are examples of events with negative offsets.

Table 3. Table of Related Events

Event ID	Schedule	Event Desc	Duration	Relation Type	Related To	Offset	Dependent Events
1	1	Event 1.0	9				[2, 3, 4, 5]
2	1	Event 1.1	2	ES	1	0	
3	1	Event 1.2	4	SS	1	4	
4	1	Event 1.3	2	EE	1	1	[8]
5	2	Event 2.a	2	SS	1	1	[6, 7]
6	2	Event 2.b	2	SE	5	-1	
7	2	Event 2.c	4	SE	5	2	
8	3	Event 3.a	2	ES	4	-1	[9, 10]
9	3	Event 3.b	2	SS	8	-3	
10	3	Event 3.c	2	SS	8	-2	

D. ICALENDAR EXTENSIONS

A reexamination of Figure 22 with more specificity for the types of properties needed to extend the capabilities advocated thus far is shown in Figure 26. The properties for each object are listed in two parts. The first part lists the iCalendar properties that are part of the existing specification that are essential and necessary for the Schedule and Relational Event objects. The second listing is of properties that do not exist in the current iCalendar specification, but are necessary to support the requirements laid out in Chapter III. These properties begin with an “X-,” which is the iCalendar format for nonstandard properties.

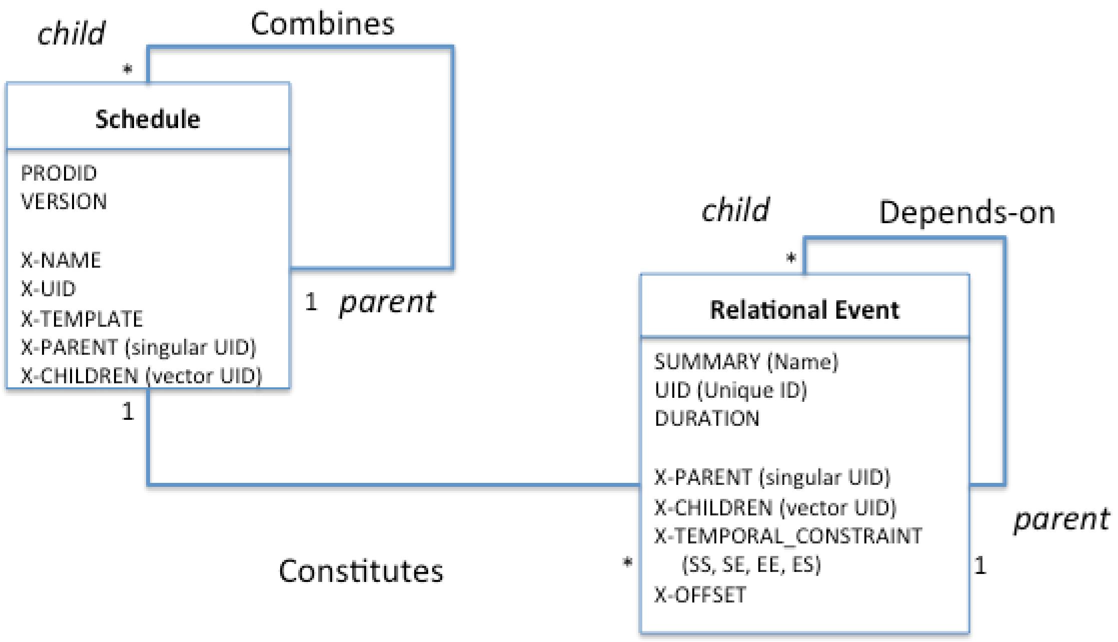


Figure 26. Property Extensions for Schedule and Relational Event Objects

1. Schedule Properties

The CALENDAR object specified in [9] has very few properties available to it. In fact, a property to give a calendar a name is nonstandard and must be specified with an "X-." Hence, we include X-NAME as a property that is necessary, especially since there will potentially be multiple schedules branched from a single schedule. The Unique Identifier (UID) in [9] is defined as a persistent globally unique identifier. Though a mandatory property for EVENT, JOURNAL, FREE BUSY, and TO DO calendar components, this is not a property belonging to the CALENDAR object. In order to tie a schedule (establish a parent/child relationship) to another, a way to reference them to each other with unique names is essential.

The X-TEMPLATE property is of type Boolean and indicates whether the schedule is a template or not. This property must be consistent throughout the schedule hierarchy. Template schedules must be recognized and treated differently than calendars or active schedules because they do not contain any reference to real dates or times. Thus, a calendar application that tried to apply a template to a real calendar would not have any point to anchor from.

Finally, the X-PARENT and X-CHILDREN properties are necessary to trace the relationships between related schedules. A singular UID in the X-PARENT property would be the parent schedule for that particular schedule. The parent schedule, in turn, would have a list of UIDs in the X-CHILDREN property for those schedules that belonged to the parent. There is no restriction to whether a schedule can have both X-PARENT and X-CHILDREN values. In a three-tiered organizational tree, the middle-tier organization would have both a parent and child(s).

2. Relational Event Properties

Similar to the X-PARENT and X-CHILDREN properties for a SCHEDULE object, these properties in the Relational Event object contain the UID for those events that have dependencies. An event that has many other events triggering off of it, will have a list of their UIDs in the X-CHILDREN property list. Conversely, each of those dependent events will have the key event's UID in their X-PARENT property field. As with schedules, there are no limitations to having values for both X-PARENT and X-CHILDREN properties.

For those dependent events (with X-PARENT property), the type of temporal relationship (discussed earlier in the chapter) must be specified in the X-TEMPORAL CONSTRAINT property. The type values would be one of the following:

1) SS–the start of the dependent event depends on the start of the parent event
2) SE–the start of the dependent event depends on the end of the parent event
3) EE–the end of the dependent event depends on the end of the parent event
4) ES–the end of the dependent event depends on the start of the parent event

In addition to specifying the X-TEMPORAL CONSTRAINT property for dependent events, the X-OFFSET property must be specified to indicate the amount of time separating the two dependent events given the temporal relationship. The type value for this is Duration (not the same as the DURATION property). As specified in [9], Duration values (prefaced with a P) are given in weeks (W), days (D), hours (H), minutes

(M), and seconds (S), and can be either positive or negative. So for example, a negative offset (duration) of 15 days, 5 hours, and 20 seconds would be: *-P15DT5H0M20S*.

V. COMPARATIVE ANALYSIS

A. METHODOLOGY

To decide whether or not these new calendaring capabilities and extensions would be beneficial and useful, we examine the efficiency and degree of error potential of each method using a simplified fictional case study that should illustrate the reasons why these capabilities would be useful and should give a determination of how useful they can be.

The selected use cases presented in Chapter III are used as the basis for action that needs to be applied to the case study. The number of steps required performing the action in the use case measures the systems efficiency. The number of potential errors surrounding the update is used to measure the error rate.

This chapter material assumes a generic calendaring system with the capabilities and features outlined in Chapter II, and assumes a future calendaring system with the extended capabilities and features outlined in Chapter III.

B. CASE STUDY

The following example will give context to the use cases presented, and serve to illustrate the deficiencies in today's mainstream calendaring applications. Consider a simple weeklong computer science conference that will be scheduled the first week of August 2011. The NPS Computer Science (CS) Department Planning Committee (PC) organizes the conference. They create and retain full authority for the master calendar event, an event lasting from Monday thru Friday. Other calendars are also created for each CS subdepartment: Artificial Intelligence (AI), Information Assurance (IA), Network Engineering (NE), and Software Engineering (SE). Each subdepartment's calendar is used to schedule their respectively hosted events such as seminars, meetings, and programming sessions. Master events, such as the kickoff meeting, guest speaker lunches, and after hour social events are planned and scheduled by the PC on the master calendar. The completed conference schedule is shown in Figure 27.

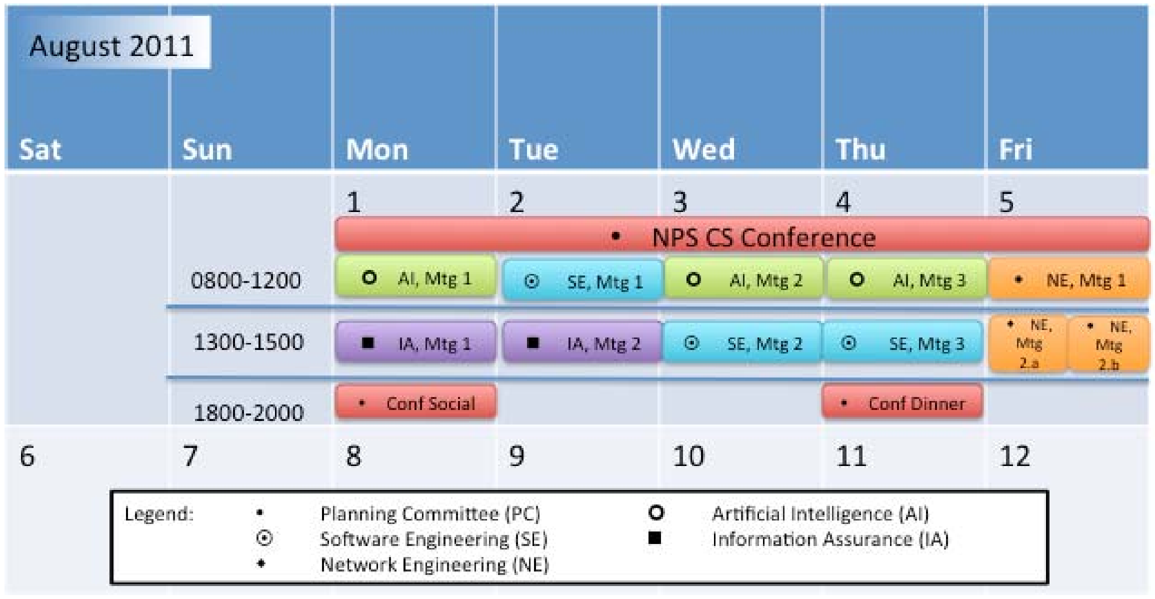


Figure 27. CS Conference Calendar

For the purpose of the comparison, we assume the calendar structure shown in Figure 27 has been already created: one conference calendar using existing tools and methods, and another conference calendar using a calendar application with enhancements described in Chapters III and IV. Although in some cases there may be more than one way to model the conference (i.e., multiple calendars can be created and shared by one individual or may be delegated separately for creation and sharing), only the method described should be considered and used for the comparison.

The degree of complexity in which the conference can be modeled can also be a point of debate. Since the conference is relatively simple and straightforward, a single calendar (vice the five in this example) can be used to model the conference. We purposely organize the conference into multiple calendars to mimic how larger and more complex projects would most likely be organized. In this way, also, we are able to show how the proposed tools simplify and aid in efficiency of certain actions, as well as better maintain the cohesiveness of the entire calendar structure.

The summary of the structure in terms of the calendar, schedule, events, and their relationships are described below.

1. Calendar Structure–Current Calendaring Capability

The following description of the methods and resulting calendar structure is used to describe the CS Conference (as it appears in Figure 27) using a mainstream calendaring system with features and functionality outlined in Chapter II. This description is the baseline of the calendar prior to any changes suggested further in the later part of the comparison.

a. Calendars

A total of five separate calendars would exist for each of the respective computer science departments: PC, AI, IA, NE, and SE. For simplicity's sake, it is assumed that one person from the PC creates, names, sets permissions, and shares each of the five calendars. From the system point of view, each of these calendars has no relation between one another. They have a peer relationship within the calendar application. As a consequence, in order for other users to see the conference (when shared to them), all five calendars need to be added to that user's calendar program.

b. Events

A total of 14 events are shown in Figure 27. Each event exists in one of the five calendars as outlined in the legend of Figure 27. There are no peer-to-peer relationships between any of the events. The only relationship that exists with an event is that of its parent container (calendar). Table 4 shows the conceptual organization and relationship (or lack thereof) of the differing conference events.

Table 4. Table of Conference Events and their Relationships (Current Calendaring)

Event	Event Number	Calendar Contained In	Relationships
NPS CS Conference	1	Planning Committee (PC)	None
Conf Social	2	Planning Committee (PC)	None
Conf Dinner	3	Planning Committee (PC)	None
AI, Mtg 1	4	Artificial Intelligence (AI)	None
AI, Mtg 2	5	Artificial Intelligence (AI)	None
AI, Mtg 3	6	Artificial Intelligence (AI)	None
SE, Mtg 1	7	Software Engineering (SE)	None
SE, Mtg 2	8	Software Engineering (SE)	None
SE, Mtg 3	9	Software Engineering (SE)	None
NE, Mtg 1	10	Network Engineering (NE)	None
NE, Mtg 2.a	11	Network Engineering (NE)	None
NE, Mtg 2.b	12	Network Engineering (NE)	None
IA, Mtg 1	13	Information Assurance (IA)	None
IA, Mtg 2	14	Information Assurance (IA)	None

2. Calendar Structure–Future Calendaring Capability

Using the ideas presented in Chapters III and IV, the following methods and structure would be used to model the CS conference schedule in Figure 27, incorporating hierarchical schedules and temporally related events.

a. Calendars (Schedules)

Five schedules would be created for each of the respective computer science departments: PC, AI, IA, NE, and SE. However, in this case, the structure and relation of these schedules differ from the flat structure of the calendars using today's calendaring methods. Each of the subdepartments schedules is created subordinate to the PC schedule. Again, we assume one person from the PC creates, names, and sets permissions for each of the five schedules. Now, because of conference schedule's hierarchical structure, however, the creator needs only to publish the PC calendar. Users who have the conference schedule shared to them need only subscribe to this single schedule. The subordinate schedule structure will also be part of the top-level schedule.

b. Events

Table 5 shows the same 14 events as in Table 4. The difference, however, is that each of the events has a Start to Start (SS) relationship with Event 1 (except Event 1 itself). We do not describe the exact details of these relationships. Each relationship would contain an offset from Event 1. For example, the event *AI Mtg 1* would likely have a 0-hour offset since it is the first event of the conference. Event *AI Mtg 2* on the other hand would likely have a 48-hour offset. These relationship details are not relevant to the comparison, so we omit them.

Table 5. Table of Conference Events and their Relationships (Future Calendaring)

Event	**Event Number**	**Schedule Contained In**	**Relationships**
NPS CS Conference	1	Planning Committee (PC)	None
Conf Social	2	Planning Committee (PC)	SS - Event 1
Conf Dinner	3	Planning Committee (PC)	SS - Event 1
AI, Mtg 1	4	Artificial Intelligence (AI)	SS - Event 1
AI, Mtg 2	5	Artificial Intelligence (AI)	SS - Event 1
AI, Mtg 3	6	Artificial Intelligence (AI)	SS - Event 1
SE, Mtg 1	7	Software Engineering (SE)	SS - Event 1
SE, Mtg 2	8	Software Engineering (SE)	SS - Event 1
SE, Mtg 3	9	Software Engineering (SE)	SS - Event 1
NE, Mtg 1	10	Network Engineering (NE)	SS - Event 1
NE, Mtg 2.a	11	Network Engineering (NE)	SS - Event 1
NE, Mtg 2.b	12	Network Engineering (NE)	SS - Event 1
IA, Mtg 1	13	Information Assurance (IA)	SS - Event 1
IA, Mtg 2	14	Information Assurance (IA)	SS - Event 1

We now present two use cases to provide the essence of how the proposed system will differ from existing calendaring systems. They will also demonstrate the applicability and usefulness of these new capabilities in organizational planning and project management.

C. APPLIED USE CASES

1. Project Timeline Shift (UC1.5)

For UC 1.5, the conference is delayed or moved ahead due to some circumstance and consequently the conference schedule needs to be adjusted in the calendar application. For this particular scenario, we choose to delay the conference by two months, starting 03 October 2011. The internal structure of the conference remains the same. Only the conference window changes, shifting from 1–5 August 2011 to 3–7 October 2011. We choose this two-month delay to better illustrate the difficulty when moving events between months. Moving events within a month, week, or day is far more trivial and less beneficial as a comparative scenario.

a. Current Calendaring Method

Doing so with today's calendaring would require each event to be manipulated to reflect the new dates (the time properties would not need to be changed if the original structure is preserved). Some calendars have the flexibility to drag-and-drop events on the calendar slate to effect the change. The schedule change, using the drag-and-drop method, can only be implemented within the scope of the current view, however. If examining an event within the daily view of the calendar, the drag-and-drop would change the time of the event. Similarly, if dragging-and-dropping the event within the monthly view, the change would affect the date property. Drag-and-drop across months is not possible or necessarily desirable. Cut-and-paste methods are also a possibility; however, this method can result in extensive navigation (e.g., moving a group of events ahead 3 months would require multiple trips back and forth between the differing months for each cut-and-paste) to affect schedule changes.

Both of the previous two methods discussed are not desirable under the current scenario. In addition, there is no feature implemented in current calendaring tools that allow for selection of multiple events, with subsequent drag-and-drop capability. Instead, the method used to implement this scenario is the direct manipulation of each individual event's start and end properties.

To effect the change to the conference schedule using this method, we must edit the properties of each event in the conference schedule. We do this using a double-click for each event. Subsequently, we make adjustments to the start and end properties for each event, followed by a "save" or "done" click to complete the change for the event. Since there are 14 total events, using this method we would need to make 14 double-clicks, and the same number of start and end adjustments, as well as the same number of "save" or "done" clicks. In total, (14 x 4) = 56 actions are required to make this change to the conference schedule using the technique just described. The summary of results is listed in Table 6.

While this method is not too difficult if the number of events is few, it could be a monumental and error-prone task if the number of events is large. With each event manipulation, there exist two possible chances to readjust the new start and end properties incorrectly. In a sea of events, these potential mistakes in start and end properties might not be easily detected once they are made. Errors to the date adjustments are much more likely in scenarios where the offset from the base event varies considerably. For this example, since the conference is only five days long, the determination of the adjusted start and end dates is relatively easy. For project schedules that span weeks or months, or where the project may be delayed by hours instead of days, the determination of adjusted start and end dates will be much more difficult.

b. Future Calendaring Method

With the conference schedule built using temporally related events (Table 5), the two-month delay can be accomplished in a small fraction of steps compared to what is required using the methodology described previously. Essentially, what would be needed from the user perspective is the adjustment to the base event (Event 1 – NPS CS Conference). The action would require a double-click, followed by the adjustment to start and end properties for this event, concluded with a "save" or "done" click.

Once completed, there may possibly be a system response and required user feedback such as "Do you want to adjust all related events (Yes/No)?" If *yes* is selected, the system would automatically update the subordinate events based on the new base event properties and the relationship specified. Note any subordinate relations would also be automatically adjusted if they existed. If *no* is selected, the system would likely be designed to adjust the offset property for each relation specified in Table 5.

Table 6. UC 1.5 – Project Timeline Shift Comparative Effort

Method	**Number of**				
	Double-Clicks	**Start Property Changes**	**End Property Changes**	**"Save" Clicks**	**Total Actions**
Current	14	14	14	14	56
Future	1	1	1	1	4

c. Comparison of Results

Table 6 shows the comparison between the two methods for shifting the timeline discussed previously. In general, the increased difficulty to perform such a project schedule adjustment is proportional to the number of events that are part of the overall schedule. In this particular case, there are 14 events. It can be seen that it is 14 times the effort to make the shift in the conference schedule with today's calendaring tools compared to the project schedule built with event-to-event temporal relationships.

The area of most concern, however, is the proportional amount of additional effort required to change the start and end properties using current tools and the methodology presented. This example shows that there is a proportionally increased opportunity to make errors for the start and end properties using current methods compared to the future calendaring capability method. The number of date changes required for the future calendaring method is two, whereas the number of date changes required using current tools is 28. There is a 14-fold increase in the inadvertent entry of a wrong date with today's calendaring tools.

2. Reuse Archived or Existing Schedule (UC3.6)

In this scenario, the annual conference is being planned, but this time it is for the following year, 2012. It is assumed that conference schedule generally worked fine the past year and can be used as a template for the current year. The conference schedule in Figure 27 exists on paper but has not yet been entered into the calendar application for August 2012. The end result should appear like the calendar shown in Figure 28. We follow the same method and assumptions outlined in the previous use case, and provide an abbreviated discussion for this use case.

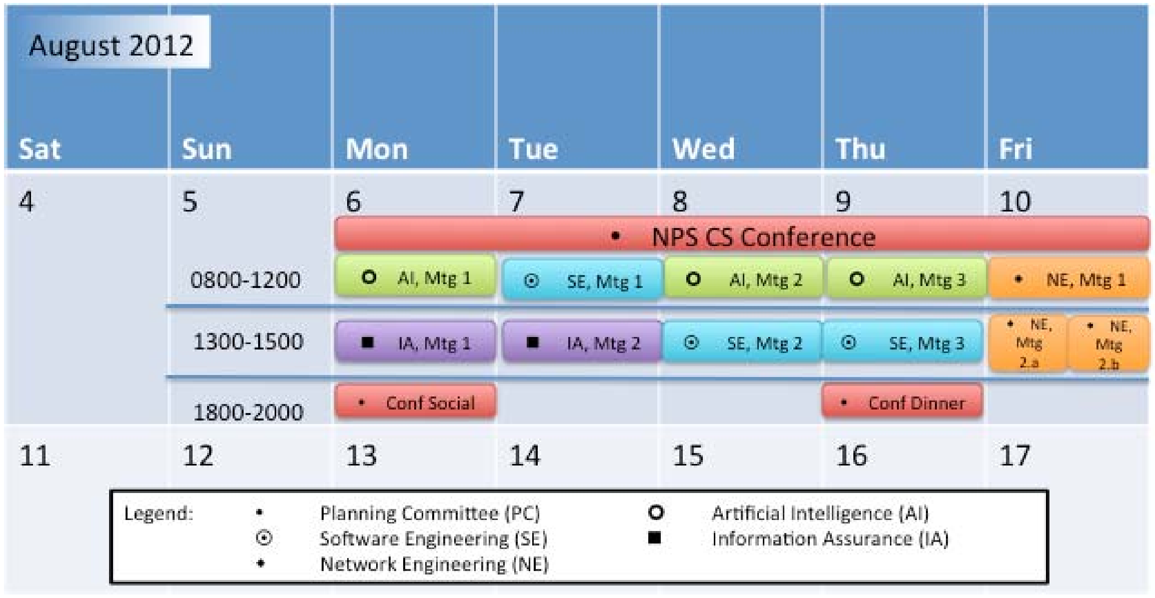


Figure 28. CS Conference Calendar (2012)

a. Current Calendaring Method

The method for creating the conference schedule depicted in Figure 28 requires the creation of the five calendars (PC, SE, NE, AI, and IA). Next, each of the 14 events will need to be created from scratch. Each created event will have to have the event name, calendar it belongs to, start, and end time specified at minimum.

b. Future Calendaring Method

Once again, the conference has previously been created using temporal events. The conference can be reused in one of two ways. First, the previous year's conference calendar may exist in the calendar. This calendar can be reused following the same method described previously for the Project Timeline Shift (UC 1.5) use case. Alternatively, the conference calendar can be "imported" from a previously saved .ics file. We use the latter case for comparison.

Table 7. UC 3.6 – Reuse Archived or Existing Schedule Effort

Method	**Number of**						
	Double -Clicks	**Descriptio n Changes**	**Parent Calendar Id**	**Start Property Changes**	**End Property Changes**	**"Save" Clicks**	**Total Actions**
Current	14	14	14	14	14	14	84
Future	1	1	0	1	1	1	5

c. Comparison of Results

Table 7 shows the disparity between the two methods. The current method requires everything to be redone from scratch. Although it is possible the 2011 conference schedule exists in the calendar database, it is as likely to exist in another form or format other than in the calendar application. A printed copy, or one belonging as part of an electronic document (i.e., slide show presentation) is likely to be a source of the previous year's conference information. Therefore, it is likely that the conference schedule will need to be put back into the calendar entirely from scratch.

The future method requires a magnitude less in terms of effort. Although the additional work of finding the .ics file in a file directory is required (and not captured in the table), this omission does not detract from the disparity between the two methods.

VI. CONCLUSION

A. SUMMARY OF WORK

This thesis examined the current state of computer calendaring with respect to modeling series event scheduling and hierarchical schedules. With the exception of project management software, we showed that calendaring software does not have the capability to adequately model complicated organizational schedules. We argued that most users, who want this type of project management scheduling capability, but not the investment in buying and using it, are forced to use other tools (i.e., tabular tools such as Excel) because calendaring tools are too limited.

As a result, we provided a description of a system that would meet these needs. We described the stakeholders who would use and benefit from such a system, described the actors that would interact with it, and provided its main features and nonfunctional requirements. We also presented a variety of use cases to describe how these features would be used and the actions the user may take as part of the given use case.

We then provided a framework for design of such a system. We did so by first analyzing and describing the essential details of the temporal relation concepts that would be needed. A domain model for current calendar users was described and contrasted against that of organizational calendar users. Also, the current calendaring conceptual data model was presented and contrasted against a data model that would support the proposed calendaring capabilities. A set of extensions to the iCalendar data format was outlined to demonstrate the relative feasibility of such a calendaring system.

Two use cases were examined to highlight the benefits of these proposed calendaring capabilities. In each case, the potential savings of human processing effort and reduction in data-entry errors are considerable. It is realized that the comparative analysis does not factor in total effort. Certainly, it requires work to create the temporally related events in the future calendaring system. However, once that work is complete, the benefits shown for the two highlighted use cases speak for themselves.

B. RECOMMENDATIONS FOR FUTURE WORK

The following are ideas where future work can be done.

1. Prototype

Implementation of hierarchical schedules and temporally related events in a calendar application will be no trivial task. A database will be required to track the events, their schedule, and their relations. Graphical properties and Human Computer Interaction (HCI) will also need to be thought out carefully. These types of event relations are not trivial to think of and implement in themselves. Any additional complexity or clutter added by the application will certainly doom its use. Such a system will, however, help tighten the specifications and extensions proposed.

2. XML Schema

The iCalendar specification will likely succumb to its XML counterpart XCal. XCal will likely be the future for calendar data exchange and interoperability. This work only proposes extensions to the iCalendar data model. These extensions, and their refined definition, will need to be stated in XML terms. Ultimately, however, if proved adoptable and desirable, these extensions will need to be part of the specification itself so that these scheduling relationships are portable across products and systems.

The nature of XML will also potentially allow these scheduling relationships to interact "smartly" with other scheduling properties. One such area of research and development is in optimization scheduling. Whereas the temporal properties addressed in this work are specified (i.e., the calendar system needs to be able to determine or is provided with fixed start and end times), other optimization scheduling systems may be able to parse and compute optimized event schedules based on a no-earlier-than or no-later-than relationship.

APPENDIX. LIST OF USE CASES

A. UC 1.1–CREATE RELATIONAL EVENT

Use Case Section	Comment
Number	1.1
Name	Create Relational Event
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	Parent Event exists.
Success Guarantee	New Event is created and visible. Relationship between New (child) Event and Parent Event is established and visible ('show relationships' == on).
Main Success Scenarios	1. User creates a new event in whatever manner offered by the application (i.e., double click on day/time, file menu selection, shortcut key, etc.). 2. Required information filled out (i.e., name) 3. In lieu of specifying the start/end (from/to) properties, user selects the button called 'has relation' or 'is related to'. 4. A list of events (relatable to the present calendar selected) is displayed (either the main calendar window itself or an agenda-style list). The user selects the Parent Event from the list. If this temporal relation is the Parent Event's first, then the Parent Event must be flagged as a 'parent'. 5. The user selects the child-parent relationship (SS, SE, ES, or EE). 6. User specifies the offset (interval of time) from the parent event. 7. User specifies the duration of the New Event, or (alternatively) the user specifies the start (if relationship is tied to the new event's end time) or end (if relationship is tied to new event's start time). In the later case, the duration can be calculated and set based on the information given. 8. User is presented with a default schedule however user can choose other available schedule. Note however only the events within a hierarchy can be related. 9. User presses the 'Done' button and the New Event is shown on the calendar along with the relationship to the Parent Event (if 'show relationships' toggled on).
Extensions	*a. At any time prior to the user selecting 'Done', the 'Cancel' button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting 'Done', all information will be lost.

B. UC 1.2–MODIFY STATIC EVENT TO RELATIONAL EVENT

Use Case Section	Comment
Number	1.2
Name	Modify Static Relational Event
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	Event to be modified exists and does not have a relation. Parent Event exists.
Success Guarantee	Relationship between Modified Event and Parent Event is established and visible (‘show relationships’ == on).
Main Success Scenarios	1. User selects desired event to be modified and selects ‘edit’. 2. User selects the button called ‘has relation’ or ‘is related to’. 3. Start and End properties are no longer editable (greyed out). 4. A list of events (relatable to the present calendar selected) is displayed (either the main calendar window itself or an agenda-style list). The user selects the parent event from the list. If this temporal relation is the parent event’s first, then the parent event must be flagged as a ‘parent’. 5. The user selects the child-parent relationship (SS, SE, ES, or EE). 6. User specifies the offset (interval of time) from the parent event. 7. User specifies the duration of the new event, or (alternatively) the user specifies the start (if relationship is tied to the new event’s end time) or end (if relationship is tied to new event’s start time). In the later case, the duration can be calculated and set based on the information given. 8. If user changes the schedule (calendar) that the event belongs to, then the allowable schedules to be chosen have to be in the hierarchy of the original schedule. 9. User presses the ‘Done’ button and the New event is shown on the calendar along with the relationship to the Parent Event (if ‘show relationships’ toggled on).
Extensions	*a. At any time prior to the user selecting ‘Done’, the ‘Cancel’ button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting ‘Done’, all information will be lost.

C. UC 1.3–MODIFY RELATIONAL EVENT TO STATIC EVENT

Use Case Section	Comment
Number	1.3
Name	Modify Relational Event to Static Event
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	Event to be modified exists and has a relation to its parent.
Success Guarantee	Relationship between modified event and parent event is dis-established. Event becomes static and any pointers from parent to child or child to parent are removed.
Main Success Scenarios	Three potential scenarios lead to this use case: 1. User selects desired event to be modified and selects 'edit'. 1.1. User selects the 'remove relation' button. 1.2. Start and End properties are set according to the previous relation to the parent event. 1.3. Modified Event is converted to a Static Event. 1.4. Pointers from the Parent Event to the Modified Event are removed. If no other child relationship for parent exists, Parent Event becomes Static Event as well. 1.5. User makes other modifications as necessary to the Modified Event. 1.6. If user changes the schedule (calendar) that the event belongs to, then any schedule within the calendar application can be chosen. 1.7. User presses the 'Done' button and the Modified Event is shown on the calendar. 2. User selects the visible relation between Child and Parent Events and subsequently 'deletes' the relationship. User is prompted to confirm delete operation. Steps 1.2 – 1.4 from above will be accomplished. 3. The Parent Event is deleted (see UC1.7).
Extensions	*a. At any time prior to the user selecting 'Done', the 'Cancel' button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting 'Done', all information will be lost.

D. UC 1.4–MODIFY TEMPORAL RELATIONSHIP BETWEEN PARENT/CHILD EVENTS

Use Case Section	Comment
Number	1.4
Name	Modify Temporal Relationship Between Parent/Child Events
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	Both Parent, Child, and the temporal relation between the two exist.
Success Guarantee	Modified Event is updated with new relationship type or offset (i.e., SS changed to SE and offset changed from 1 hr to 2 hr).
Main Success Scenarios	There are two main scenarios in which this can occur: 1. User selects desired event to be modified and selects 'edit'. 1.1. User may elect to do one or both of the following: a. User modifies the offset (i.e., 1 day:1 hour changed to 2 days: 2 hours). b. User changes child-parent relationship (i.e., SS changed to SE). 1.2. User makes any other modification to event if needed. 1.3. User presses the 'Done' button. 1.4. System saves new offset and/or child-parent relationship with the Child Event. 1.5. System displays calendar with Child and Parent Events showing their new relationship (if 'show relationships' toggled on). 2. User graphically changes the offset and or parent/child relationship. 2.1. User may elect to do one or both of the following: a. User drags Child Event to the desired time location in the Calendar. System will prompt user to confirm the new offset. b. User clicks on ends of relationship arrow (on the visible link between Parent/Child Events) and reassigns them as necessary to the Start or End of either the Parent or Child Events. 2.2. Items 1.4 and 1.5 above are executed by the System.
Extensions	*a. At any time prior to the user selecting 'Done', the 'Cancel' button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting 'Done', all information will be lost.

E. UC 1.5–MODIFY PARENT TEMPORAL PROPERTIES (MOVE TIMELINE)

Use Case Section	Comment
Number	1.5
Name	Modify Parent Temporal Properties (Move Timeline)
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	Parent and Children Events exist.
Success Guarantee	The Start or End properties of a Parent Event are modified. All children with temporal relations to the changed properties are readjusted on the calendar.
Main Success Scenarios	There are two main scenarios in which this can occur: 1. User selects desired Parent Event to be modified and selects ‘edit’. 1.1. User modifies the Start and/or End Property with a new date/time. 1.2. User presses the ‘Done’ button. 1.3. System saves new information for Parent Event and calls for a redisplay of the calendar. 1.4. System calendar shows the adjusted Parent Event and displays the shifted Children Events (Note, the child properties do not change). 2. User graphically changes the Parent Event either by clicking and dragging the Event to a new date/time, or by ‘grabbing’ the Start or End edges of the Parent Event and expanding or shrinking the event accordingly. 2.1 Actions 1.3 and 1.4 from above are executed and calendar is re-displayed with the shift in schedule.
Extensions	*a. At any time prior to the user selecting ‘Done’, the ‘Cancel’ button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting ‘Done’, all information will be lost.

F. UC 1.6–DELETE TEMPORAL EVENT (CHILD)

Use Case Section	Comment
Number	1.6
Name	Delete Temporal Event (Child)
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	A Child Event exists with a temporal relation to its Parent Event. In this case, the Child Event does not have any children of its own.
Success Guarantee	The Child Event is deleted. Any reference to the deleted event on the parent side is removed as well.
Main Success Scenarios	1. User selects desired Child Event and selects ‘delete’. 2. System prompts user for confirmation. 3. User confirms deletion. 4. System removes any pointers on Parent Side. 5. System removes Child Event from storage. 6. System calls for a redisplay.
Extensions	*a. At any time prior to the user selecting ‘Done’, the ‘Cancel’ button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting ‘Done’, all information will be lost.

G. UC 1.7–DELETE TEMPORAL EVENT (PARENT)

Use Case Section	Comment
Number	1.7
Name	Delete Temporal Event (Parent)
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	A Parent Event exists with its children.
Success Guarantee	The Parent Event is deleted. All children of the parent are converted to static events by default. User can elect to delete all related events. If so children are deleted from the data store. Pointers to the Parent Event from another parent are also removed.
Main Success Scenarios	1. User selects desired Parent Event and selects ‘delete’. 2. System prompts user for confirmation. 3. User confirms deletion. 4. System prompts user if user wishes to delete all subordinate children: 4.a. User chooses not to delete all children (default). 4.a.1. If deleted event has parent, system removes pointer from parent. 4.a.2. For each pointer to a child event, System converts the child event to a static event. 4.a.3. Event in question is deleted from data store. 4.a.4. System calls for a calendar redisplay. 4.b. User chooses to delete all children 4.b.1. Item 4.a.1 is performed. 4.b.2. Each child event of the event in question is deleted. Before deletion though, if this child event has children of its own, the process of 4.b is performed on the child event. 4.b.3. Items 4.a.3 and 4.a.4 are performed.
Extensions	*a. At any time prior to the user selecting ‘Done’, the ‘Cancel’ button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting ‘Done’, all information will be lost.

H. UC 2.1–TOGGLE SELECTION OF VISUAL RELATIONSHIPS

Use Case Section	Comment
Number	2.1
Name	Toggle Selection of Visual Relationships
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	None
Success Guarantee	The visual representation of the temporal relationships between events are either shown or not shown depending on whether or not the visual representation feature is turned on or off respectively.
Main Success Scenarios	1. User selects on / off of visual representation icon, shortcut, or menu item. 2. If the user selects on: 2.a. System calls for a calendar redisplay with visual relationships shown (if any). 3. If the user selects off: 3.a. System calls for a calendar redisplay with visual relationships not shown.
Extensions	None

I. UC 2.2–GANTT CHART REPRESENTATION OF PROJECT

Use Case Section	Comment
Number	2.2
Name	Gantt Chart Representation of Project
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	At least one calendar (schedule) is selected
Success Guarantee	The selected calendars (schedules) to be viewed are displayed in a Gantt chart form.
Main Success Scenarios	1. User selects Gantt chart view. 2. System provides Gantt chart view of selected calendars (schedules). 3. A vertical scroll bar is necessary if too many calendars (schedules) are selected for view. The scroll bar will allow user to scroll vertically through all the different calendars (schedules). 4. A horizontal scroll bar will be displayed if project window does not fit on the screen. Scrolling right will show future events. Scrolling left will show past events.
Extensions	*a. At any time prior to the user selecting 'Done', the 'Cancel' button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting 'Done', all information will be lost.

J. UC 2.3–MATRIX VIEW OF PROJECT

Use Case Section	Comment
Number	2.3
Name	Matrix View of Project
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	A schedule is available to be shown in the matrix view.
Success Guarantee	The schedule is displayed in a tabular (matrix) form on the main display of the calendaring tool.
Main Success Scenarios	1. User selects schedule to be displayed (most likely through right click of the schedule). 2. User selects matrix view. 3. System places matrix view of schedule in the main window. 4. System places items in chronological order row by row. 5. System provides vertical scroll bar if the number of rows (events) exceeds the screen display. 6. System provides horizontal scroll bar if the number of event properties needing display exceeds the screen display.
Extensions	*a. At any time prior to the user selecting ‘Done’, the ‘Cancel’ button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting ‘Done’, all information will be lost.

K. UC 2.4–SELECTIVE VIEWING OF SCHEDULES IN HIERARCHY

Use Case Section	Comment
Number	2.4
Name	Selective Viewing of Schedules in Hierarchy
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	At least one calendar (schedule) is available for display.
Success Guarantee	The appropriate calendars (schedules) are displayed on the calendar based on user desires.
Main Success Scenarios	1. User deselects calendar (schedule) in the pane showing all available calendars and schedules. 2. If a schedule is deselected, all subordinate schedules are also deselected. 3. If a schedule is re-selected, it and all its subordinate schedules are displayed. 4. System commands redisplay of selected or deselected calendars to main calendar view.
Extensions	*a. At any time prior to the user selecting ‘Done’, the ‘Cancel’ button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting ‘Done’, all information will be lost.

L. UC 3.1–CREATE NEW CHILD SCHEDULE

Use Case Section	Comment
Number	3.1
Name	Create New Child Schedule
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	At least one schedule is available for display.
Success Guarantee	The new schedule is created and placed underneath selected schedule.
Main Success Scenarios	1. User selects schedule (most likely through right click on the schedule) in calendar display pane. 2. User selects "create new schedule" from pop-up menu. 3. System creates new schedule subordinate to the selected schedule. System assigns unique color to new schedule. 4. User provides name for new schedule.
	*a. At any time prior to the user selecting 'Done', the 'Cancel' button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting 'Done', all information will be lost.

M. UC 3.2–MOVE SCHEDULE WITHIN THE HIERARCHY

Use Case Section	Comment
Number	3.2
Name	Move Schedule Within the Hierarchy
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	A multiple schedules exist within a single schedule. At least two levels of hierarchy exist within the schedule.
Success Guarantee	The appropriate schedule is relocated to a new parent. Any subordinate schedules are also moved.
Main Success Scenarios	1. User selects schedule (2nd level calendar or lower) in calendar display pane. 2. User drags selected schedule and moves it to new parent within existing hierarchy. 3. System makes appropriate changes to relationship table and the data store. 4. System provides appropriate schedule structure changes in the calendar display pane.
Extensions	*a. At any time prior to the user selecting 'Done', the 'Cancel' button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting 'Done', all information will be lost.

N. UC 3.3–FREEZE SCHEDULE

Use Case Section	Comment
Number	3.3
Name	Freeze Schedule
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	A schedule exists in the calendar application.
Success Guarantee	Users are prevented from making additions, deletions, or changes to the schedule during the freeze.
Main Success Scenarios	1. User (owner) selects schedule and freezes schedule. 2. System updates schedule properties to reflect freeze. Subordinate schedules are also frozen. 3. System provides visual notification to users that schedule is frozen. 4. System prevents other calendar users from making changes to schedule. 5. Freeze is released when user (owner) releases freeze.
Extensions	*a. At any time prior to the user selecting 'Done', the 'Cancel' button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting 'Done', all information will be lost.

O. UC 3.4–DELETE SCHEDULE (PROJECT CANCELLATION)

Use Case Section	Comment
Number	3.4
Name	Delete Schedule (Project Cancellation)
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	Schedule to be deleted exists in calendar application.
Success Guarantee	The schedule, its subordinate schedules, and all associated events are removed from display and calendar data store.
Main Success Scenarios	1. User selects schedule to be deleted and deletes schedule. 2. System prompts user if they are sure of deletion. If sure: 3. System removes the schedule, subordinate schedules, and associated events from the display. 4. System removes the schedule, subordinate schedules, and associated events from the data store.
Extensions	*a. At any time prior to the user selecting ‘Done’, the ‘Cancel’ button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting ‘Done’, all information will be lost.

P. UC 3.5–SAVE SCHEDULE (PROJECT ARCHIVAL)

Use Case Section	Comment
Number	3.5
Name	Save Schedule (Project Archival)
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	A schedule exists in the calendar application.
Success Guarantee	The schedule to be saved, its subordinate schedules, and all associated events are “exported” and saved to a file.
Main Success Scenarios	1. User selects the schedule to be saved and selects export. 2. The system prompts the user with a file system menu and prompts user to name the file to be saved and its location. 3. Users provides information. 4. System produces exported file (in accordance with RFC 5545 and extensions proposed in Chapter IV) and saves file to location.
Extensions	*a. At any time prior to the user selecting ‘Done’, the ‘Cancel’ button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting ‘Done’, all information will be lost.

Q. UC 3.6–REUSE ARCHIVED OR EXISTING SCHEDULE

Use Case Section	Comment
Number	3.6
Name	Reuse Archived or Existing Schedule
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	Either the schedule to be duplicated exists in the calendar data store or it exists as a file (see UC 3.5).
Success Guarantee	The reused calendar is given a new name and exists in the calendar data store likely with new base date.
Main Success Scenarios	There are two main scenarios in which this can occur: 1. User selects schedule to be reused from the calendar view pane. 1.1. User selects 'copy' 1.2. User selects 'paste' in the appropriate location in the appropriate location in the calendar view pane. 1.3. System provides new / unique color scheme. 1.4. User provides unique name for schedule. 1.5. System displays schedule information on the calendar. 1.6. User makes changes to base event dates to reflect new project window. 2. User selects import from the menu bar. 2.1. System provides file system menu. 2.2. User selects appropriate file to be imported. 2.3. System imports schedule to the data store and displays schedule information on the calendar. 2.4. User makes changes to base event dates to reflect new project window.
Extensions	*a. At any time prior to the user selecting 'Done', the 'Cancel' button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting 'Done', all information will be lost.

R. UC 3.7–PUBLISH (SHARE) / SUBSCRIBE SCHEDULE

Use Case Section	Comment
Number	3.7
Name	Publish (Share) / Subscribe Schedule
Scope	Calendar Application
Level	User Goal
Primary Actor	User
Preconditions	At least one schedule is available to share / subscribe to. Schedule has be located on central data store and not on a singular computer.
Success Guarantee	The schedule is made available to other users. Users are able to bring schedule into their calendar application for display.
Main Success Scenarios	1. User (sharer) selects schedule to be shared. 2. User (sharer) sets appropriate share properties and grants user access. 3. System makes appropriate updates to the data store. 4. User subscribes to schedule made available to them. 5. System provides authentication. 6. System transfers schedule data to the subscriber's calendar application. 7. Subscriber's calendar application displays subscribed schedule in calendar.
Extensions	*a. At any time prior to the user selecting 'Done', the 'Cancel' button is selected and all information pertaining to the new event will be discarded. *b. If system fails prior to the user selecting 'Done', all information will be lost.

LIST OF REFERENCES

[1] R. Pagan, "POA&M 2005 BRAC Corrections Implementation Plan," HQMC Plans Policy and Operations, Security Division, unpublished, Jul 2010.

[2] 3rd Marine Aircraft Wing, "Operational Deployment Milestones," USMC, unpublished, May 2005.

[3] M. A. Trabun, "U.S. Marine Corps Training Modeling and Simulation Master Plan," USMC Training and Education Command, Technology Division, File 18 Jan 2007.

[4] K. R. Baker and D. Trietsch, *Principles of Sequencing and Scheduling*. Hoboken, NJ: Wiley & Sons, 2009.

[5] J. P. Lewis, *Fundamentals of Project Management*, 3rd ed. New York, NY: Amacom Books, 2007.

[6] H. Kerzner, *Project Management, A Systems Approach to Planning, Scheduling, and Controlling*, 10th ed. Hoboken, NJ: Wiley & Sons, 2009.

[7] WPF calendar schedule control, DevComponents™ [Online]. Available http://www.devcomponents.com/dotnetbar-wpf/WPFScheduleControl.aspx.

[8] Module 3: Calendar > Chapter 4: Create and view calendars, GoogleApps™ [Online]. Available http://edutraining.googleapps.com/Training-Home/module-3-calendar/chapter-4/2-7.

[9] B. Desruisseaux, *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*, IETF RFC 5545, September 2009; [Online]. Available http://tools.ietf.org/html/rfc5545.

[10] C. Daboo, *iCalendar Transport-Independent Interoperability Protocol (iTIP)*, IETF RFC 5546, December 2009 [Online]. Available http://tools.ietf.org/html/rfc5546.

[11] A. Melnikov, *iCalendar Message-Based Interoperability Protocol (iMIP)*, IETF RFC 6047, December 2010; http://tools.ietf.org/html/rfc6047.

[12] C. Daboo, *xCal: The XML Format for iCalendar*, IETF RFC 6321, 2011.

[13] An introduction to Internet Calendaring and Scheduling, CalConnect [Online]. Available http://www.calconnect.org/CD1012_Intro_Calendaring_V1.1.shtml.

[14] T. Considine, *WS-Calendar Version 1.0*, OASIS Committee Specification 01, 30 July 2011 [Online]. Available: http://docs.oasis-open.org/ws-calendar/ws-calendar/v1.0/ws-calendar-1.0-spec.html.

[15] Use cases and requirements, CalConnect [Online]. Available http://www.calconnect.org/usecases.shtml.

[16] C. Larman, *Applying UML and Patterns*, 2nd ed. Upper Saddle River, NJ: Prentice Hall Professional, 2002.

[17] PMO and project management dictionary, PM Hut [Online]. Available http://www.pmhut.com/pmo-and-project-management-dictionary.

[18] Planning and resource management, Penn State University [Online]. Available: https://courses.worldcampus.psu.edu/welcome/pmangt/samplecontent/520lesson08/lesson08_02.html.

[19] K. Neumann et al., *Project Scheduling with Time Windows and Scarce Resources*. Berlin, Heidelberg: Springer-Verlag, 2002.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California

7. Professor Tom Otani
   Naval Postgraduate School
   Monterey, California

8. Professor Man-Tak Shing
   Naval Postgraduate School
   Monterey, California